VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MULTIMEDIÁLNÍ DIFF - AUDIO DOKUMENTY

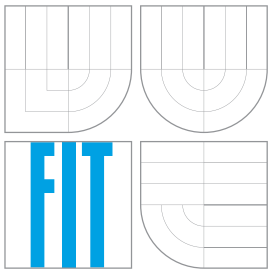
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KOMADEL

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MULTIMEDIÁLNÍ DIFF - AUDIO DOKUMENTY

MULTIMEDIA DIFF - AUDIO DOCUMENTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KOMADEL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR CHMELAŘ

BRNO 2011

Abstrakt

Tato práce se zabývá vývojem nástroje, který umožňuje funkci diff nad audio soubory obsahujícími obecný zvuk jako hudbu, řeč a jiné zvuky. Jsou zde uvedeny poznatky z různých oblastí vědy zabývající se zvuky jako například psychoakustiky, zpracování řeči a automatické kategorizace hudby podle žánru. Jsou zde popsány některé algoritmy diffu a také externí nástroje potřebné pro vývoj cílové aplikace. Dále tato práce uvádí návrh a implementaci výsledné aplikace, nastavení použitá pro extrakci charakteristických znaků zvuku a zhodnocení dosažených výsledků.

Abstract

This work describes development of a diff tool working with audio files containing general sound such as music, speech and other sounds. There are presented facts from different domains of science related to sound, such as psychoacoustics, speech recognition and automatic music genre categorisation. This paper also contains description of some diff algorithms and external tools needed for development of the goal application. Moreover, there is introduced design and implementation of the application, settings used for sound features extraction and evaluation of attained results.

Klíčová slova

diff, audio dokument, cepstrum mel-frekvence, SVD, HTK.

Keywords

diff, audio document, mel-frequency cepstrum, SVD, HTK.

Citace

Michal Komadel: Multimediální diff - audio dokumenty, diplomová práce, Brno, FIT VUT v Brně, 2011

Multimediální diff - audio dokumenty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Chmelaře.

. . . . . .
Michal Komadel
24.5.2011

Poděkování

Týmto děkuji mému vedoucímu Ing. Petru Chmelaři za cenné rady a vedení počas tvorby této práce.

Obsah

Kapitola 1

Úvod

Úlohou tohoto projektu je vytvorenie nástroja na porovnávanie digitálnych záznamov zvuku (hudby, reči), ktorý by mal pracovať podobne ako všeobecne známy unixový diff pre textové dokumenty. Tento nástroj bude súčasťou (pluginom) väčšieho projektu s názvom MediaDiff, ktorý bude schopný zaručiť funkciu diff pre rôzne typy dokumentov.

V praxi je unixový diff notoricky známy avšak pracuje len s textovými dokumentami, prípadne umožňuje porovnanie po bytoch. V rôznych oblastiach ako rozpoznávanie reči, žánru hudby, prípadne pri vyhľadávaní audia v multimediálnych databázach, by podobný nástroj, ktorý pracuje s audio záznamami, mohol pomôcť a výrazne uľahčiť prácu.

Želaným výsledkom tejto práce je teda aplikácia, ktorý porovná zvukové záznamy a dokáže rozoznať, ktoré časti sú zhodné, pridané, či pozmenené, samozrejme tak aby prípadné rozdiely bolo možné počuť ľudským uchom. Keďže nie všetky zvuky dokáže človek vnímať, je treba brať ohľad na parametre a obmedzenia ľudského ucha a princípy spracovania zvuku človekom.

Pre tento spôsob porovnávania existuje hneď niekoľko dôvodov. Napríklad ak má užívateľ dve vzorky rovnakého audia pričom v jednej vzorke je implementovaný audio vodoznak (samozrejme nepočuteľný), je nutné, aby tento nástroj vyhodnotil audio vzorky ako rovnaké. Alebo v prípade ak má audio vzorku uloženú vo formáte s bezstratovou kompresiou a novú vzorku vytvorí použitím stratovej kompresie (napríklad MPEG s rozumnou bitovou rýchlosťou, tak aby poslucháč nepočul rozdiel), tento nástroj musí znovu vyhodnotiť obe vzorky ako rovnaké.

Napriek tomu, že existuje množstvo informácií najmä z oblasti rozpoznávania reči, prípadne automatického rozpoznávania žánru hudby, informácie o metrikách, ktoré pracujú rovnako dobre pre hlas aj hudbu (či iné zvuky) rovnako spoľahlivo, neexistujú. Z tohoto dôvodu bude nutné pri vývoji takejto aplikácie skombinovať znalosti z oblasti psychoakustiky, rozpoznávania reči a iných oblastí zaoberajúcich sa hudbou a zvukmi, vymyslieť experimenty na základe ktorých, budú nastavené najvhodnejšie parametre aplikácie a nakoniec overiť funkčnosť a spoľahlivosť vytvoreného nástroja.

Ďalšou nutnosťou je implementovať vhodný algoritmus diffu, pretože bude nutné porovnávať veľké množstvo dát a keďže tento nástroj má byť vhodný pre bežné použitie, je nutné, aby spracovanie a vyhodnotenie zvukových záznamov trvalo rozumnú dobu vzhľadom na veľkosť porovnávaných audio záznamov.

V druhej kapitole tejto práce sú uvedené niektoré poznatky z psychoakustiky a vlastnosti ľudského sluchového aparátu, ktoré okrem toho, že sa používajú (nielen) pri stratovej kompresii audia, budú brané do úvahy aj pri implementácii cieľového nástroja pre diff zvuku. Taktiež je tu uvedený spôsob digitalizácie reálnych audio signálov a spôsob ich uloženia na

pamäťovom médiu.

Tretia kapitola predstavuje niektoré typy charakteristických rysov zvuku používaných v oblasti rozpoznávania reči, či iných oblastiach zaoberajúcich sa rozpoznávaním reči alebo klasifikáciou hudby a tiež spôsob efektívneho zmenšenia objemu získaných dát.

Štvrtá kapitola predstavuje niekoľko algoritmov diffu s rôznymi priestorovými a časovými vlastnosťami. V implementácii cieľového nástroja diffu bude použitý variant jedného z týchto algoritmov.

Ďalšia kapitola popisuje externé nástroje, ktoré budú potrebné pre určité oblasti implementácie a tiež programovacie jazyky, v ktorých je výsledná aplikácia implementovaná.

Podrobný popis návrhu a samotného spôsobu implementácie celého nástroja, je uvedený v kapitole 6. Testy, ktoré pomohli určiť vhodné nastavenia extrakcie vybranej reprezentácie zvuku, sú uvedené v kapitole 7. V nasledujúcej kapitole sú uvedené výsledky testov cieľovej aplikácie pre rôzne typy porovnávaných záznamov.

Posledná kapitola obsahuje zhodnotenie práce a výsledkov testov. Uvádza nedostatky implementovanej aplikácie a možné vylepšenia, na ktoré by sa bolo vhodné zamerať v ďalších verziách tohoto nástroja.

Kapitola 2

Zvuk

V tejto časti definujem čo je zvuk a uvediem niektoré princípy a obmedzenia spracovania zvuku ľudským uchom a mozgom, ktoré sa s úspechom využívajú v stratovej kompresii audia ako napríklad MPEG-1 [37]. Tieto obmedzenia budú musieť byť vzaté do úvahy aj pri implementácii cieľového nástroja, aby bolo možné určiť, ktorý zvuk je počuteľný ľudským uchom. V prvej časti sú popísané vlastnosti sluchového aparátu odvíjajúce sa od frekvencie zvuku, v druhej časti vlastnosti od jeho intenzity a času dopadu zvukovej vlny na ušný bubienok.

Najprv je ale vhodné definovať, čo to vlastne zvuk je. Zvuk je pozdĺžne vlnenie, ktoré sa šíri cez sled kompresií a riedenia v médiu, zvyčajne vzduchu [19]. V psychoakustike sú zvuky zvyčajne popisované pomocou tlaku, hustoty, frekvencie a šírkou pásma zvuku [10].

2.1 Limit frekvencie zvuku a absolútny prah počutia

Ľudské ucho dokáže zachytiť zvukové vlny v rozmedzí 20Hz až 20kHz. S rastúcim vekom sa horná hranica rapídne znižuje a v dvadsiatich rokoch klesne až na 16kHz, v šesťdesiatich sa pohybuje na hodnote približne 10kHz. Navyše ľudský sluch nie je veľmi citlivý na zvuky s frekvenciou nižšou ako 100Hz [19, 10, 7]. Tento fakt sa využíva v známom a často používanom stratovom kódovaní audia MPEG-1. Väčšina MPEG-1 kodérov odstráni z audia všetky zvuky z frekvenciou vyššou ako 16kHz [34].

Absolútny prah počutia udáva minimálnu intenzitu (úroveň tlaku zvuku) nutnú k tomu, aby určitý tón bol počuteľný v bezhlučnom prostredí. Bolo zistené, že táto intenzita vo všeobecnosti závisí od frekvencie tónu [37]. Pri nízkych frekvenciách je prah na pomerne vysokej úrovni (50Hz - 40dB) a s rastúcou frekvenciou klesá (200Hz – 15dB). Pre zvuky s frekvenciou medzi 2000Hz a 5000Hz je hodnota prahu najnižšia, dokonca nižšia ako 0dB. Následne s rastúcou frekvenciou rastie aj absolútny prah počutia. Do 12kHz sa pohybuje v rozmedzí 0dB až 15dB, potom začína rapídne rásť a pri 16kHz nie zvuk počuteľný ani pri veľmi vysokej hodnote jeho intenzity (samozrejme v závislosti od veku osoby). Z týchto faktov sa dá usudzovať, že ľudský sluch najintenzívnejšie reaguje na zvuky s frekvenciou o hodnote medzi 2000Hz a 5000Hz aj pri veľmi nízkych hodnotách jeho intenzity [10].

2.2 Maskovanie a dočasné okno integrácie

Najprv je nutné si uviesť pojem kritické pásmo. Na základe znalostí fyziológie ľudského sluchového systému existuje predpoklad, že ľudské ucho spracúva zvukové signály v závis-

losti od ich frekvencie. Existujú dôkazy podporujúce predpoklad, že ľudské ucho pracuje ako pásmová priepusť a každé pásmo je do určitej miery spracúvané samostatne. Tieto experimentálne určené frekvenčné pásma sú nazývané kritické pásma [37]. V rámci tohoto pásma môže dôjsť k maskovaniu jedného tichšieho tónu druhým hlasnejším. To znamená, že jeden z týchto tónov nebude vnímaný. Pokiaľ jednotlivé frekvencie tónov nebudú spadať do rovnakého kritického pásma k maskovaniu nedôjde. Napríklad hlasný vysoký tón nezabráni počutiu tichého basu. Kritické šírky začínajú na veľkosti 50Hz až 100Hz na spodnej časti frekvenčného rozsahu ľudského ucha a zostávajú takto úzke niekoľko kHz. Následne sa začne rozsah kritických pásiem logaritmicky zväčšovať [8].

Existujú v princípe dva druhy maskovania, a to simultánne a nesimultánne maskovanie. Simultánne maskovanie nastáva vo chvíli ak dva alebo viac zvukov dosiahnu ušný bubienok v rovnakom čase. Experimenty ukázali existenciu prahov maskovania závisiacich na frekvencii maskujúceho zvuku. Ak intenzita zvuku je pod príslušným prahom, tento zvuk bude zamaskovaný a nebude vnímaný. Aby tieto prahy mohli byť určené, je nutné identifikovať zvuk buď ako tón alebo ako šum, pretože boli nájdené rôzne úrovne prahov v závislosti na tom, či maskujúci zvuk je tón alebo šum [37].

V sluchovom systéme je tiež možné aj nesimultánne maskovanie, ktoré nastáva ak krátko trvajúci zvuk zamaskuje zvuk, ktorý nasleduje po ňom (postmasking), prípadne zamaskuje zvuk, ktorý ho predchádza (premasking). V prvom prípade časový rozdiel zvukov môže byť v intervale 50ms až 300ms, v druhom prípade musí byť časový rozdiel medzi 1ms až 2ms. Toto maskovanie je taktiež závislé na frekvencii, intenzite a dĺžke trvania maskujúceho zvuku [37].

Dočasné okno integrácie (temporal window of integration) je mechanizmus mozgu, ktorý zvuky zaznamenané sluchovým aparátom v určitom, krátkom, časovom rozmedzí integruje do jediného zvukového vnemu. Dĺžka tohoto okna je približne 160ms až 170ms [21].

2.3 Digitalizácia signálu zvuku

Z definície zvuku vieme (viď vyššie), že je to pozdĺžne vlnenie, čiže signál v reálnom svete, ktorý môže byť v každom čase charakterizovaný určitými vlastnosťami. Tento signál je spojitý v čase a v amplitúde. Spojitým signálom sa hovorí tiež analógové, pretože zmeny týchto signálov v čase sú analogické k zmenám zdroja signálu [39].

Keďže počítačové systémy sú diskrétneho charakteru, nie je možné priamo zaznamenať spojitý(analógový) signál na pamäťové médium. Aby mohol byť tento signál uložený na diskrétne médium, je nutné ho najprv diskretizovať. Najjednoduchší spôsob diskretizácie je vzorkovanie spojitého signálu v čase. Aby pri vzorkovaní signálu nedochádzalo k zmenám obsahu v zaznamenávanom diskrétnom signále a mohol byť neskôr správne reprodukovaný (nesmie dôjsť k aliasu), je nutné aby vzorkovanie spĺňalo Shanonov vzorkovací teorém. Tento teorém hovorí o tom, že vzorkovacia frekvencia musí byť najmenej dva krát väčšia ako frekvencia vzorkovaného signálu. Takéto vzorkovanie sa nazýva Nyquistovo vzorkovanie [24, 42].

Štandardným systémom pre bezstratovú diskretizáciu audia je PCM (Pulse Code Modulation). Tento systém je vhodný pre diskretizáciu všeobecných analógových signálov, avšak čo sa týka audia je to štandard pre zaznamenávanie audia na audio CD (Red Book Audio). PCM vzorkuje signál v pravidelných intervaloch a v týchto intervaloch nastáva zmena signálu len jedným smerom. Veľkosť vzorky je úmerná napätiu audio signálu a táto vzorka je reprezentovaná celým číslom. Po diskretizácii pomocou PCM je analógový audio signál zaznamenaný vo forme poľa celých čísiel (vzoriek), kde poradie vzorky je analógia k času

a veľkosť tejto vzorky je analógiou k veľkosti tlaku zvuku na mikrofón v danom čase. Keďže mikrofóny (prípadne iné zvuk zaznamenávajúce zariadenia) majú vždy obmedzené frekvenčné pásmo aj zaznamenaný audio signál bude frekvenčne obmedzený [40, 11].

PCM audio používa viacero vzorkovacích rýchlostí (bežné sú 22.05kHz, 44.1kHz, 48kHz, 96kHz) a tiež viacero rozlíšení pre vzorky (veľkosť vzoriek v bitoch). Bežné rozlíšenia pre vzorku sú 8b, 16b, 24b. PCM formát je bežným formátom pre AIFF a WAV. Stratové formáty ako MPEG-1 a AAC vznikajú stratovou kompresiou bezstratových formátov (PCM), využitím nedokonalostí ľudského sluchového aparátu (viď kapitolu 2), uložením týchto dát do rámcov a pridaním ďalších meta dát ako napríklad ID3 tagy [11].

DC (direct current) zložka signálu sa v digitálnom spracovaní signálov (nielen audia) týka hodnoty ofsetu, ktorá sa v čase nemení. Tento DC ofset môže byť do audio signálu vložený analógovo-digitálnym konvertorom. Napríklad ak máme signál x a príslušný výstup y ofset (DC zložka) by vyzeral približne takto: $y(t) = x(t) + C$, kde C je konštantné reálne číslo. Toto jednoducho spôsobí posunutie audio signálu nahor (alebo nadol) bez zmeny tohoto signálu. V ideálnom prípade nie je možné počuť rozdiel medzi signálom s DC zložkou a bez nej, avšak táto zložka môže v určitých prípadoch spôsobiť poškodenie signálu. Toto je možné z dôvodu konečného počtu bitov použitých pre reprezentáciu vzorky signálu v digitálnych systémoch. Pri veľkej hodnote DC zložky je možné, že časť amplitúdy signálu bude odrezaná, pretože veľkosť amplitúdy spolu z veľkosťou DC zložky bude väčšia ako maximálna veľkosť zaznamenateľná na danom počte bitov. V praxi je vhodné túto zložku odstrániť už pred vKrátkodobá funkcia energiezorkovaním signálu alebo v prípade analýzy už navzorkovaného signálu túto zložku odstrániť vo fáze predspracovania. Toto môže byť uskutočnené pomocou použitia filtrov alebo vypočítaním aritmetického priemeru signálu a následným odstránením tohoto priemeru z daného signálu [31].

Kapitola 3

Extrakcia charakteristických rysov zvuku

V kapitole 2 sme sa dozvedeli akým spôsobom je audio uložené na diskovom médiu v počítačových systémoch. Avšak prosté porovnávanie vzoriek audio záznamov nie je vhodné v prípade ak chceme pri tomto porovnávaní vziať do úvahy aj obmedzenia ľudského sluchového aparátu uvedené v kapitole 2. Z tohoto dôvodu je nutné z audio dát extrahovať ďalšie údaje, ktoré nám zaručia analógiu týchto obmedzení. Pri implementácii nástroja audio diffu, bude na základe týchto údajov rozhodnuté, aké charakteristické rysy budú extrahované.

V úvode tejto kapitoly budú vysvetlené niektoré pojmy z oblasti spracovania signálov a psychoakustiky nutné k pochopeniu niektorých krokov postupu pri extrakcii charakteristických rysov z audio signálu. V nasledujúcej časti sú uvedené fakty o niektorých rysov extrahovaných z audio signálu používaných v oblastiach spracovania reči a klasifikácie audia podľa jeho obsahu a tiež postupy ako tieto charakteristické znaky z jednotlivých audio záznamov extrahovať.

3.1 DFT, DCT, Windowing a Mel mierka

DFT je skratka pre diskrétnu Fourierovu transformáciu. DFT sa používa pri riešení širokého spektra problémov vo vede a technike. Je to matematický nástroj, ktorý umožňuje upravenie problému tak, aby mohol byť oveľa ľahšie riešiteľný. V podstate Fourierova transformácia rozloží alebo oddelí signál (vlnu) na sinusoidy s rozličnými frekvenciami, z ktorých pozostáva pôvodný signál. Určuje alebo rozdeľuje sínusoidy podľa ich frekvencie a príslušnej amplitúdy.

Pretože počítačové systémy majú diskrétny charakter, používa sa diskrétna Fourierova transformácia, ktorá vyžaduje diskrétny signál (zodpovedá spôsobu zaznamenávania signálov vzorkovaním) a ako výstup počíta diskrétne vzorky transformácie [18].

DCT je diskrétna kosínusová transformácia. DCT je špeciálny prípad DFT, pri ktorej sa eliminujú sínusové komponenty koeficientov a výsledkom je len jediné číslo. Toto sa dosiahne jednoducho opakovaním vstupných vzoriek v obrátenom poradí a opakovaním DFT nad množinou vzoriek s dvojnásobnou dĺžkou. Dvojnásobná dĺžka množiny vzoriek sa získa pridaním zrkadlovo obrátených vstupných vzoriek. Dôvodom pridania zrkadlovo obrátených vzoriek vstupného signálu je zmena tohoto signálu na párnu funkciu, ktorej sínusové koeficienty sú rovné nule. DCT vytvorí rovnaký počet koeficientov ako vstupných vzoriek. DCT je tiež primárne používané v audio kódovaní, pretože konvertuje vstupný signál do formy, kde redundancia dokáže byť jednoducho identifikovaná a odstránená [40].

V prípade ak potrebujeme analyzovať signál, zvyčajne ho neanalyzujme vzorku po vzorke a taktiež pokiaľ ide o väčší signál nie je vo väčšine prípadov možné ho analyzovať ako celok. Analýza pomocou vzoriek by zahŕňala príliš veľa detailov a naopak pri analýze signálu ako celku by sa mohlo veľa informácií stratiť. Vo všeobecnosti je pri analýze signálu určitá skupina vzoriek ohraničená oknom a následne analyzovaná. Technicky je signál vnútri okna prenásobený hodnotou okna v danom mieste. Táto metóda zamerania sa a extrakcie určitého množstva sekvenčných vzoriek sa nazýva windowing. Existuje niekoľko používaných okien, z ktorých najjednoduchšie je obdĺžnikové okno.

Obdĺžnikové okno je charakteristické ostrými okrajmi. Toto okno má jednotkový hodnotu v rámci hraníc okna a mimo toto okno má hodnotu rovnú nule. Pri použití tohoto okna je signál v rámci okna prenásobený hodnotou jedna a mimo tohoto okna hodnotou nula. Takýmto spôsobom extrahujeme signál s nezmenenou amplitúdou, ktorý sa nachádza v medziach okna. Toto okno kvôli jeho tvaru spôsobuje zotrvačnosť frekvencie, avšak je vhodné len pre signály charakteristické rýchlou zmenou.

Ďalším známym a používaným oknom je Hannovo okno. Implementácia tohoto okna je dosiahnutá pomocou posunutia a sčítania Dirichletových jadier. Hannovo okno je tiež niekedy nazývané kosínusové okno.

Asi najčastejšie používané okno je Hammingovo okno, ktoré je podobné Hannovmu s určitými modifikáciami pri váhovaní Dirichletových jadier. Dôležitou charakteristikou tohoto okna je, že na krajoch má nenulové hodnoty, a preto je niekedy nazývané napoly zdvihnuté kosínusové okno. Správa sa o čosi viac podobne obdĺžnikovému oknu ako Hannovo okno a má lepšie krátkodobé vlastnosti. Je často používané v spektrálnej analýze a syntéze audia [31].

Mel mierka je pokus o nájdenie spôsobu merania psychologického zážitku z výšky tónov. Existuje niekoľko mel mierok na základe výskumov s poslucháčmi. Napríklad v roku 1940 Stevens a Volkman požiadali poslucháčov, aby rozdelili rozsah zvuku od 200Hz do 6500Hz na štyri rovnaké intervaly. Tento experiment ukázal zhustenie vnímaného rozsahu nad frekvenciou 1000Hz. Mel mierka má vždy menší rast ako hudobná stupnica a rastie monotónne do frekvencie 5kHz. Z tohoto dôvodu je interval jednej oktávy pri nízkej frekvencii vnímaný oveľa menší ako pri vyššej frekvencii. Hoci táto mierka bola kritizovaná z hudobného ale aj psychoakustického hľadiska, zachytáva, že rozdiel vo vnímaní zvuku o frekvencii medzi 1000Hz a 2000Hz je výrazne väčší ako rozdiel vo vnímaní zvuku s frekvenciou 100Hz a 200Hz [16].

3.2 Koeficienty cepstra mel-frekvencie

Koeficienty cepstra mel-frekvencie (ďalej len MFC koeficienty) sú efektívnym prostriedkom k reprezentácii krátkodobej frekvenčnej charakteristiky zvukového signálu, zohľadňujúce vnímanie ľudského sluchového aparátu. Sú založené na nelineárnej kosínusovej transformácii logaritmického energetického spektra na nelineárnej Mel mierke frekvencie. Kompenzujú nerovnomerné vnímanie ľudského sluchového aparátu naprieč frekvenčným spektrom. MFC koeficienty sú široko používané viac ako 30 rokov v oblasti rozpoznávania reči a v poslednej dobe tiež v oblasti klasifikácie zvuku (hudba, zvuky okolia, audio prenos správ) [9, 38, 23, 43, 22].

Pri extrahovaní MFC koeficientov z audio signálu sa najprv prúd vzoriek audio signálu rozdelí na časti. Tieto časti sú typicky nazývané rámcami a tieto rámce sa získajú procesom windowing, pričom sa zvyčajne (pri spracovaní reči) používa Hannove alebo Hammingovo okno o veľkosti 20ms až 30ms, pričom posuv okna nastáva každých 10ms. Z toho vyplýva, že

tieto okná sa prekrývajú. Hannove alebo Hammingovo okno sa používa z dôvodu stlmenia nesúvislostí na okrajoch okna.

Následne sa aplikuje diskrétna Fourierova transformácia nad vzorkami z každého rámca a je získané frekvenčné spektrum pre každý rámec. Toto lineárne frekvenčné spektrum je potom deformované na Mel mierku frekvencie, čo zodpovedá nelineárnemu vnímaniu zvuku sluchovým aparátom človeka. Takto pozmenené spektrum je prevzorkované (počet nových vzoriek je nižší ako pôvodný) pomocou použitia skupiny trojuholníkových filtrov s typicky 24 až 40 kanálmi, ktorých frekvenčná šírka sa mení podľa mel mierky. Výpočet hodnoty koeficientov tejto skupiny filtrov pozostáva z prenásobenisa každej hodnoty Fourierovej transformácie s príslušným ziskom trojuholníkového filtru a následným súčtom takto vypočítaných hodnôt. Tieto hodnoty sa potom logaritmicky zhustia, aby bol modelovaný vzťah medzi intenzitou zvuku a jeho vnímanou hlasitosťou. Výsledné MFCC získame pomocou diskrétnej kosínusovej transformácie zhustených hodnôt [9, 43, 6].

Podľa [38] sa pre rozpoznávanie reči typicky používa 13 MFC koeficientov, ale z experimentov v [38] zistili, že pre klasifikáciu žánru hudby je najdôležitejších prvých 5 koeficientov. Prvé MFC koeficienty môžu byť tiež považované za aproximáciu spektrálnej obálky signálu [22].

V praxi môže dôjsť k situácii, že sa budú porovnávať dva audio záznamy, z ktorých jeden bude kódovaný v bezstratovom kódovaní a druhý v stratovom ako napríklad MPEG-1. V tomto prípade podľa výskumov v [14] dokonca aj pri rovnakom audio zázname, kódovanom rôznymi spôsobmi, sú extrahované MFC koeficienty rôzne. Výsledky týchto výskumov ukazujú, že MFC koeficienty nie sú závislé len na bitovej rýchlosti MPEG-1 kódovania, ale aj na akustických vlastnostiach daného audio záznamu. Očakáva sa, že tieto rozdiely by mali výrazný dopad na nástroje využívajúce MFC koeficienty, najmä ak pracujú s rôzne kódovanými audio záznamami. Tento problém bol vyriešený pomocou normalizácie MFC koeficientov. Boli skúmané tri typy normalizácie, a to normalizácia priemerom cepstra, normalizácia štandardnou odchýlkou a normalizácia priemerom a štandardnou odchýlkou [14].

Normalizácia priemerom cepstra sa získa odčítaním priemerného vektoru cepstra od pôvodného vektoru. Táto normalizácia odstráni odklon v cepstre spôsobený šumom z kanálového rušenia alebo charakteristikami mikrofónu. Táto normalizácia môže byt prevedená len "offline" čiže len na vopred nahratej vzorke a nie na online audio zázname, pretože je nutné vypočítať celkový priemerný vektor cepstra z danej audio vzorky [6, 14]. Normalizácia štandardnou odchýlkou sa vykoná vydelením originálneho vektoru cepstra štandardnou odchýlkou. Normalizácia štandardnou odchýlkou a priemerom je normalizácia vektoru cepstra vyššie uvedenými normalizáciami. Podľa výsledkov experimentov v [14] táto normalizácia výrazne priblížila získané vektory MFC koeficientov pre rovnaké audio záznamy s rôznym kódovaním. Najlepšie výsledky boli dosiahnuté s kódovaním MPEG-1 so 128kbps, kedy extrahované MFC vektory boli skoro totožné s tými extrahovanými z pôvodného nekódovaného audia [14].

V spracovaní reči sa zvyčajne pred extrakciou MFC koeficientov vykoná ešte operácia nazývaná preemfáza. Cieľom tejto operácie je kompenzovať časť vysokých frekvencií, ktoré boli potlačené počas vytvárania zvuku v ľudskom vokálnom trakte.

Okrem toho sa pri rozpoznávaní reči pomocou DTW (viď 4.1) často používa operácia nazvaná cepstrálny zdvih. Táto operácia by mala podstatne zlepšiť výsledky rozpoznávania. Cepstrálny zdvih má tendenciu potlačiť pomalé zmeny v logaritmickom spektre, ktoré väčšinou pochádzajú z pridaného bieleho šumu. Toto je dôvod prečo tento zdvih dokáže podstatne vylepšiť rozpoznávanie najmä reči obsahujúcej veľa šumu [30].

Ďalšie opatrenie je treba urobiť v prípade ak sa extrahujú MFC koeficienty z okna

ohraničujúceho signál s nulovou energiou. To by znamenalo počítanie logaritmu z nuly, ktorý má nedefinovanú hodnotu. Existuje niekoľko riešení tejto situácie. Prvé je jednoducho nulu nahradiť za veľmi malú nenulovú hodnotu. Druhý spôsob je pridanie náhodného jednobitového šumu do celého signálu. Tento proces sa nazýva dithering [9].

3.3 Ďalšie typy charakteristických znakov zvuku

Nasledujúce charakteristické znaky zvuku boli vyvinuté ako alternatíva ku koeficientom cepstra mel-frekvencie. Sú odvodené od iných vlastností zvuku, avšak nie sú tak často používané ako MFC koeficienty alebo sú poprípade použité ako doplnok k MFC koeficientom.

Krátkodobá funkcia energie charakterizuje signál v časovej doméne a poskytuje vhodnú reprezentáciu zmien amplitúdy audio signálu v čase. Pri výpočte tejto charakteristiky sa uvažuje, že sa energia audio signálu mení v krátkom intervale relatívne pomaly. Energia sa vypočíta pre okno, ktoré obsahuje určité množstvo vzoriek audio signálu. Toto okno sa pohybuje signálom (napríklad každých 10ms) a pre každý výsek určený oknom je vypočítaná príslušná energia, pričom hranice jednotlivých okien sa prekrývajú (podobne ako pri výpočte MFC koeficientov). Nevýhodou tejto charakteristiky je, že vysoké úrovne niektorých vzoriek signálu majú tendenciu posunúť celkový odhad energie, pretože jej výpočet je založený na druhej mocnine hodnoty individuálnych vzoriek signálu.

Táto charakteristika sa využíva pri spracovaní reči v prípade ak je žiadané rozlíšiť časti v audio nahrávke, ktoré obsahujú hlas, a ktoré obsahujú len ticho. Energia častí, ktoré obsahujú hlas je totiž ďaleko väčšia ako tých, ktoré obsahujú len ticho (prípadne šum). Taktiež je možné túto charakteristiku použiť k rozlíšeniu zvukov, ktoré je počuť od ticha v prípade veľkého pomeru signálu k šumu. Navyše vzor zmeny energie môže odhaliť rytmus a periodické vlastnosti zvuku [41, 33].

Ďalšou možnou reprezentáciou zvuku je krátkodobá priemerná rýchlosť prekročenia nuly. V kontexte signálov s diskrétnym časom, k prekročeniu nuly dochádza pokiaľ dve po sebe idúce vzorky majú rôzne znamienka. Táto charakteristika udáva hrubý odhad frekvenčného obsahu audio signálu. Toto je pravda najmä pri signále s úzkym frekvenčným pásmom. Hodnoty krátkodobej priemernej rýchlosti prekročenia nuly sú znovu počítané pri procese windowing, podobne ako pri výpočte krátkodobej energie, pričom sa v jednotlivých oknách počíta priemerná rýchlosť prekročenia nuly.

Model tvorenia reči predpokladá, že energia znelého hlasového signálu je sústredená pod 3kHz a energia neznelého hlasového signálu vo vyšších frekvenciách a zároveň vysoké (nízke) frekvencie naznačujú vysokú (nízku) rýchlosť prekročenia nuly, potom pre znelý hlasový signál je priemerná rýchlosť prekročenia nuly vysoká, pričom pre neznelý hlasový signál je priemerná rýchlosť prekročenia nuly nízka. To znamená, že táto charakteristika je použiteľná pre rozlíšenie znelého a neznelého hlasového signálu.

V porovnaní s hlasom má hudba oveľa menší počet zmien a priemernú amplitúdu, čo naznačuje, že priemerná rýchlosť prekročenia nuly je v čase oveľa stabilnejšia. Krivky priemernej rýchlosti prekročenia nuly majú pri hudobnom signále vo všeobecnosti nepravidelný tvar s meniacou sa základňou a relatívne malým rozsahom amplitúdy.

Keďže audio signál z prostredia môže pochádzať z veľkého množstva zdrojov, krivky priemernej rýchlosti prekročenia nuly takýchto signálov môžu mať rôzne vlastnosti. Takéto audio signály je možné klasifikovať podľa vlastností ich kriviek priemernej rýchlosti prekročenia nuly ako napríklad pravidelnosť, periodicita, stabilita a rozsah amplitúdy [41, 33].

Inou možnou charakteristikou zvuku je krátkodobá základná frekvencia. Harmonický zvuk pozostáva z radu frekvenčných zložiek, medzi ktoré patrí aj základná frekvencia a

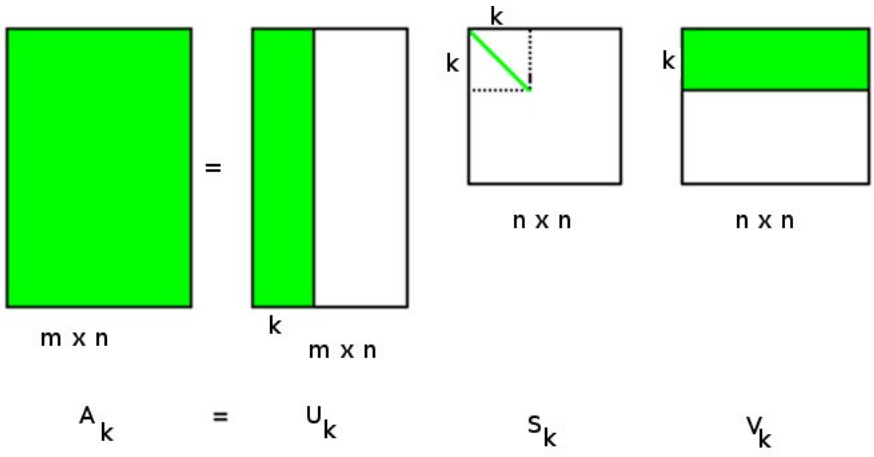
frekvencie, ktoré sú celočíselným násobkom základnej frekvencie. Takýmto spôsobom je možné rozdeliť zvuky na harmonické a neharmonické. Či je audio signál harmonický alebo nie záleží na zdroji signálu. Väčšina hudobných nástrojov vydáva harmonický zvuk a rečový signál je zmesou harmonických (znelé časti) a neharmonických (neznelé časti) zvukov. Väčšina zvukov prostredia je neharmonická.

Odhad základnej frekvencie (tónu) patrí medzi jednu z najdôležitejších otázok v spracovaní reči. Táto frekvencia nesie dôležitú prozodickú informáciu vo väčšine jazykov a navyše v tónových jazykoch sémantickú informáciu. Krátkodobá základná frekvencia je pri harmonickom audio signále rovná základnej frekvencii a pre neharmonický audio signál je krátkodobá základná frekvencia rovná nule. Hudba je zvyčajne spojite harmonická a základná frekvencia sa mení pomalšie ako pri ostatných typoch zvuku, navyše hodnota krátkodobej základnej frekvencie má tendenciu koncentrovať sa na určitých hodnotách počas krátkej doby [41, 32].

3.4 Dekompozícia na singulárne hodnoty

Dekompozícia na singulárne hodnoty (singular value decomposition – SVD) nie je technika na extrakciu charakteristických znakov z audia, avšak je využívaná v (nielen) oblasti rozpoznávania reči k redukcii dimenzionality extrahovaných dát a tiež k redukcii nežiadúceho šumu v týchto dátach. Získané dáta sú v tomto prípade reprezentované maticou. Napríklad pri extrahovaní MFC koeficientov z audio signálu jeden riadok obsahuje koeficienty získané zo signálu ohraničeného pomocou jedného okna a počet stĺpcov určuje počet extrahovaných koeficientov. Keďže v praxi by následná manipulácia z takto dimenzovanými dátami bola výpočtovo náročná, použije sa algoritmus SVD k redukcii dimenzionality. Tento algoritmus rozloží maticu A obsahujúcu dáta na tri matice U, S, V^T. U je matica, ktorej riadky sú eigen vektory matice AA^T, S je matica, ktorej hodnoty na diagonále sú singulárne hodnoty A a V je matica, ktorej stĺpce sú eigen vektory matice A^TA (V^T je transponovaná V)

Pokiaľ chceme redukovať dimenzionalitu matice A vytvoríme maticu A_k, ktorá sa nazýva k-aproximácia A tak, že ponecháme len prvých k stĺpcov U, a prvých k riadkov S a V^T. Tento postup je ilustrovaný na obrázku 3.1 [13, 5].



Obrázek 3.1: Redukcia dimenzionality pomocou SVD. Prevzaté a upravené z [5].

Kapitola 4

Porovnanie extrahovaných rysov

Po extrahovaní charakteristických znakov audia je možné pristúpiť k porovnávaniu týchto znakov, samotnému algoritmu diffu a generovaniu požadovaného výstupu. Existuje mnoho algoritmov diffu, s rôznymi vlastnosťami a keďže táto operácia je výpočtovo vysoko náročná je nutné vybrať a následne implementovať najvhodnejší a najefektívnejší algoritmus.

V tejto kapitole je v úvodnej časti najprv uvedený spôsob určovania podobnosti signálov v oblasti spracovania reči a následne uvedená všeobecná definícia diffu, jeho požadovaný výstup a základné, v praxi nepoužiteľné algoritmy slúžiace skôr pre ilustráciu podstaty tohoto algoritmu. Keďže existuje veľa prístupov a algoritmov, ktoré s rôznou časovou a priestorovou zložitosťou zabezpečujú požadovaný výstup, po úvodnej časti nasleduje predstavenie niektorých algoritmov diffu s rôznymi vlastnosťami založených na rozdielnych princípoch. Tieto algoritmy boli navrhnuté, aby pracovali s textovými súbormi, a preto je základný porovnávaný prvok riadok alebo znak. Pri ich popise teda budú používané tieto základné prvky, čo však nebráni použiť tieto algoritmy pre aj pre súbory obsahujúce iné typy dát.

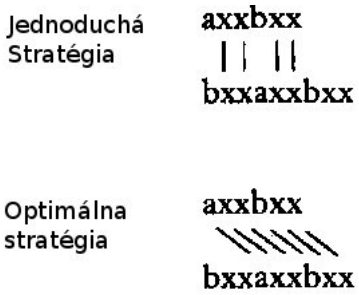
4.1 Dynamické ohýbanie času

Dynamické ohýbanie času (dynamic time warping – DTW) je prístup používaný v mnohých oblastiach ako napríklad v oblasti rozpoznávania reči pri rozpoznávaní vzoriek izolovaných slov, či porovnávanie video sekvencií a podobne. DTW je algoritmus slúžiaci k porovnaniu sekvencií, ktoré sa môžu líšiť v čase a rýchlosti. To znamená, že hľadá optimálne nelineárne zarovnanie dvoch sekvencií. Jeho priestorová a časová zložitosť je lineárna. Príkladom môže byť porovnanie dvoch rovnakých slov, avšak vyslovených rôznou rýchlosťou. Algoritmus DTW dokáže efektívne určiť podobnosti v sekvenciách týchto dvoch slov a vypočítať ich vzájomnú vzdialenosť. Na základe tejto vzdialenosti sú následne oba vzorky vyhodnotené ako rovnaké. V praxi pri rozpoznávaní slov je porovnávaná vzdialenosť získaného vzorku od vopred pripravených referenčných sekvencií uložených v databáze a hľadá sa referenčné slovo, ktorého vzdialenosť od získaného vzorku je najmenšia [26, 2].

4.2 Algoritmus diff

Úlohou algoritmu diff je nájdenie vzájomných rozdielov medzi dvoma súbormi. Výsledkom takéhoto algoritmu je v ideálnej implementácii minimálny zoznam zmien, podľa ktorého je možné zmeniť jeden súbor tak, aby bol rovnaký ako druhý. Jadrom takéhoto typu algoritmu

je zvyčajne nájdenie najdlhšej spoločnej postupnosti, z čoho vyplýva nájdenie minimálneho zoznamu zmien (minimálneho editovacieho skritpu), prípadne priame hľadanie minimálneho editovacieho skriptu. Najjednoduchšou metódou porovnávania súborov je kontrolovať súbory riadok za riadkom (prípadne znak po znaku, v závislosti od žiadaného detailu) pokiaľ sú tieto riadky rôzne, následne prechádzať riadky v súboroch, kým nie je nájdená skupina rovnakých riadkov. Nezávisle od stratégie resynchronizácie, tento prístup môže vytvoriť zoznamy zmien dlhšie ako je skutočne nutné [20, 25].



Obrázek 4.1: Porovnávanie súborov po znakoch. Prevzaté a upravené z [25].

Na obrázku 4.1 je znázornený tento postup a môžme vidieť, že takto vytvorený editovací skript obsahuje zbytočné zmeny niektorých znakov. Optimálny skript by obsahoval len pridanie prvých troch znakov z druhého súboru do prvého. Hoci v praxi v implementovaných algoritmoch diffu dochádza k resynchronizácii až po nájdení niekoľkých za sebou nasledujúcich rovnakých riadkov, žiadna takáto resynchronizačná stratégia, ktorá hľadá určitý počet po sebe nasledujúcich riadkov nie je optimálna.

Ďalšou možnosťou je využitie najdlhšej spoločnej postupnosti a napríklad použitie schémy jednoduchého algoritmu pre nájdenie najdlhšej spoločnej postupnosti z dynamického programovania. Povedzme, že riadky prvého súboru sú $A_i, i = 1..m$, druhého súboru $B_j, j = 1..n$. Ak P_{ij} je dĺžka najdlhšej postupnosti spoločnej pre prvé i-riadky prvého súboru a prvé j-riadky druhého súboru potom platí:

$$\begin{array}{ll} P_{i0}{=}0 & i = 0, ..., m \\ P_{j0}{=}0 & j = 0, ..., m \end{array}$$

$$P_{ij} = \begin{cases} 1 + P_{i-1,j-1} & \text{if } A_i{=}B_j \\ \max(P_{i-1,j}, P_{i,j-1}) & \text{if } A_i \neq B_j \end{cases} \quad 1 \leq i \leq m,\ 1 \leq j \leq m \tag{4.1}$$

Z toho vyplýva, že P_{nm} je dĺžka najdlhšej spoločnej postupnosti a z poľa P_{ij} generovaného pri hľadaní tejto postupnosti je jednoduché získať jej prvky. Po získaní prvkov najdlhšej spoločnej postupnosti C sa potom vytvorí minimálny editovací skript súčasným prechádzaním porovnávaných súborov A,B a postupnosti C a prvky nachádzajúce sa v A a chýbajúce v C označiť jedným spôsobom a označiť tie nachádzajúce sa v B a chýbajúce v C spôsobom druhým. Časová a priestorová zložitosť tohoto algoritmu je však v najhoršom prípade $O(mn)$[20]. Taktiež podľa [17] nie je táto formulácia problému vhodná pretože minimalizácia počtu pridaní a zmazaní prvkov z jedného súboru do druhého nie je najlepšie riešenie, keďže je jednoduchšie zmeny zobraziť ako pridanie, zmazanie prípadne presunutie celého bloku prvkov.

Existuje veľa rôznych algoritmov diffu, ktoré sa snažia pracovať s menšou časovou alebo priestorovou zložitosťou. Jeden z prvých bol predstavený Wagnerom a Fischerom s časovou

a priestorovou zložitosťou $O(n^2)$ a riešil problém nimi nazývaný oprava reťazcu na reťazec (string-to-string correction). Neskôr tento algoritmus vylepšil Hirschberg zavedením nájdenia najdlhšej spoločnej postupnosti v lineárnom čase.

Nástroje implementujúce algoritmus diff majú širokú škálu použitia v rôznych oblastiach nielen počítačových systémov ale aj vedy. Porovnávacie programy dokážu povedať ako a kde sa dve verzie súborov líšia. Napríklad kvôli zisteniu rozdielov medzi verziami programových modulov, vytváraniu testovacích súborov pre zmeny v module, prípadne automatickú zmenu starej verzie súboru na novú (program patch). Ďalej sa využíva pri zlučovaní nezávislých zmien súboru (trojcestné zlučovanie), kedy zmeny vzniknuté v dvoch kópiách súboru majú byť zlúčené do jedného súboru s ohľadom na pôvodný súbor. Taktiež pri častých zmenách revízií súboru je ekonomické ukladať len zmeny v súbore a nie celý nový súbor (samozrejme pokiaľ vyžadujeme možnosť návratu k starej verzii). Programy implementujúce algoritmus diff sa tiež využívajú v molekulárnej biológii, kde sú porovnávané rôzne makromolekuly pozostávajúce z reťazcu aminokyselín (proteíny) alebo nukleotidov (DNA, RNA) [27, 17, 36].

4.3 Algoritmus Hunt-McIlroy

Tento algoritmus nazývaný Hunt-McIlroy je predstavený v [20] a je použitý pre prvú implementáciu programu diff na systémoch UNIX a je podstatne rýchlejší ako tradičný algoritmus dynamického programovania [28].

4.3.1 Princíp a popis algoritmu

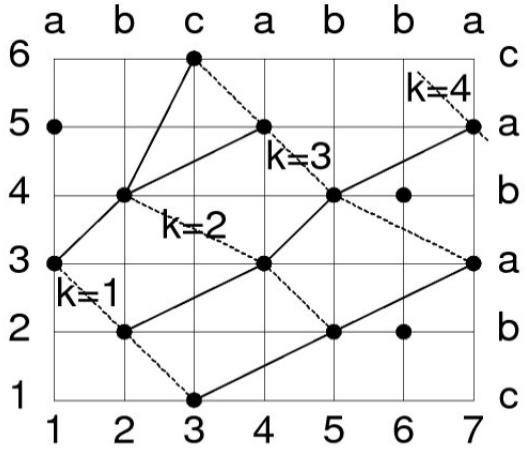
Tento algoritmus vylepšuje pôvodný algoritmus dynamického programovania tak, že sa zaoberá len podstatnými zhodami, ktorých porušenie by zmenilo najdlhšiu spoločnú postupnosť P.

Podstatné zhody, podľa Hirschberga k-kandidáti, nastanú pokiaľ $A_i = B_j$ a zároveň $P_{ij} > \max(P_{i-1}, j, P_{i,j-1})$. K kandidát je pár indexov (i, j) takých, že $A_i = B_j$ a najdlhšia spoločná postupnosť dĺžky k sa nachádza medzi prvými i prvkami prvého súboru a prvými j prvkami druhého súboru a navyše neexistuje žiadna spoločná postupnosť dĺžky k, v prípade ak sa zredukuje i alebo j. Kandidát je pár indexov, ktorý je k-kandidát pre nejaké k. Je zjavné, že hľadaná najdlhšia spoločná postupnosť sa nachádza v úplnom zozname kandidátov.

Ak (i_1, j_1) a (i_2, j_2) sú oba k-kandidáti, potom platí $i_1 < i_2$ a $j_1 > j_2$. Pokiaľ by platilo $j_1 = j_2$, (i_2, j_2) je porušená podmienka, ktorá hovorí, že neexistuje žiadna spoločná postupnosť dĺžky k, v prípade ak sa zmenší i alebo j. V prípade $j_1 < j_2$ potom spoločná postupnosť k končiacou prvkom (i_1, j_1) môže byť predĺžená na spoločnú postupnosť dĺžky $k + 1$ končiacou prvkom (i_2, j_2).

Na obrázku 4.2 môžeme vidieť jednoduchú grafickú interpretáciu určovania kandidátov. Na tomto obrázku sú bodkami znázornené body, pre ktoré platí $A_i = B_j$. Spoločná postupnosť je množina bodov pospájaná striktne rastúcou krivkou. Tieto krivky sú na obrázku uvedené a spájajú všetky body, ktoré sú kandidátmi. Hodnota k je uvedená pri každej spoločnej postupnosti a postupnosti s rovnakou hodnotou k sú pospájané bodkovanou krivkou. Tieto krivky naopak musia monotónne klesať. Kandidátov je viditeľne menej ako mn (okrem triviálnych prípadov). V praxi sa ukazuje, že týchto kandidátov je podstatne menej ako mn, takže zoznam kandidátov je možné uložiť veľmi ľahko [20].

Body na grafe sú uložené v lineárnom priestore nasledujúcim spôsobom:



Obrázek 4.2: Spoločné postupnosti a kandidáti pri porovnávaní sekvencií `abcabba` a `cbabac`. Prevzaté a upravené z [20].

1. Zostroja sa zoznamy ekvivalentných tried prvkov v druhom súbore. Tieto zoznamy zaberajú priestor o veľkosti $O(n)$ a je možné ich vytvoriť zoradením prvkov v druhom súbore,

2. každá príslušná ekvivalenčná trieda sa pridelí každému prvku z prvého súboru. Tento zoznam je uložený v priestore o veľkosti $O(m)$.

Takýmto spôsobom sme práve získali zoznam bodov na každej vertikále podobne ako na obrázku 4.2

Teraz sa začnú generovať kandidáti smerom zľava doprava. Povedzme, že K je pole najpravejších k-kandidátov doteraz navštívených pre každé k. Pre zjednodušenie výpočtu kandidátov sa toto pole vyplní zľava 0-kandidátom a do každého k, ktoré nemá ešte žiadneho kandidáta sa doplní kandidát $(m+1, n+1)$, ktorého komponenty sa pri porovnávaní vyhodnotia ako vyššie od všetkých ostatných kandidátov. K-pole je na začiatku prázdne až na výplň (0-kandidát a $(m+1, n+1)$ kandidát). Pri posune doprava sa toto pole aktualizuje. Po nájdení najdlhšej spoločnej postupnosti sa na jej základe vytvorí editovací skript rovnakým spôsobom, aký je popísaný v úvode kapitoly, pri získaní najdlhšej spoločnej postupnosti pomocou algoritmu dynamického programovania [20].

4.3.2 Efektívnosť algoritmu

Kvôli zlepšeniu efektivity pri porovnávaní veľkých súborov sa v tomto algoritme hašuje každý prvok súboru a pri porovnávaní prvkov používa hodnotu hašu. Toto však môže spôsobiť nesprávne porovnanie a vyhodnotenie niektorých rôznych prvkov ako rovnakých. To znamená, že pri použití vhodnej hašovacej funkcii je pravdepodobnosť, že nastane takéto nesprávne porovnanie $1/M$, kde M je počet možných hodnôt hašovacej funkcie. Najdlhšia spoločná postupnosť o dĺžke k je potom tvorená k prvkami určenými podľa ich hašovacích hodnôt, kde k je ďaleko menšie ako M. Potom pravdepodobnosť, že táto najdlhšia spoločná postupnosť obsahuje zle porovnaný prvok je k/M. Podľa testov v [20] bola pravdepodobnosť, že dôjde k takejto chybe pri 5000 riadkoch, menej než 10%.

Algoritmus sa chráni proti takejto chybe tak, že overí nájdenú najdlhšiu spoločnú postupnosť v pôvodných súboroch. Výsledok je potom vytvorený tak, že tieto nesprávne porovnané

prvky sú z najdlhšej spoločnej postupnosti odstránené, hoci existuje malá pravdepodobnosť, že táto najdlhšia spoločná postupnosť nie je v skutočnosti najdlhšia [20].

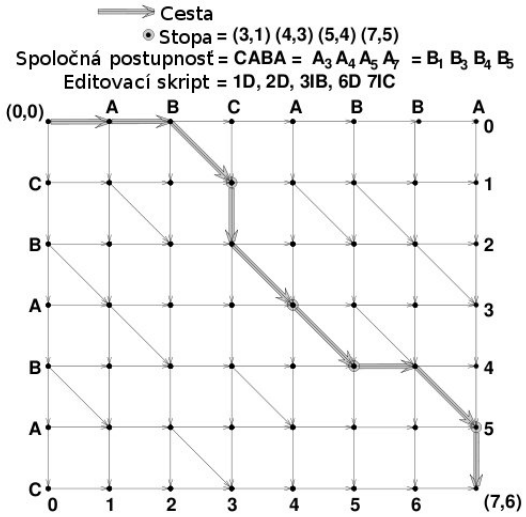
V najhoršom prípade sa tento algoritmus správa porovnateľne s algoritmom dynamického programovania. Najväčšiu časovú zložitosť má pri tom hľadanie kandidátov a následne najdlhšej spoločnej postupnosti, ktoré ma časovú zložitosť $O(mn \log m)$. Priestorová zložitosť je v najhoršom prípade $O(mn)$. V praxi však tento algoritmus pracuje značne efektívnejšie a len málokedy je nájdených viac kandidátov ako $\min(m, n)$ [20].

4.4 Myersov algoritmus

Tento algoritmus bol publikovaný v [27] a je založený na jednoduchom formalizme editovacieho grafu. Problém nájdenia najdlhšej spoločnej postupnosti transformuje na problém nájdenia najkratšej cesty v grafe. Je tiež základom pre implementáciu aktuálneho nástroja unixového diffu a v praxi by mal byť dva až štyri krát rýchlejší ako algoritmus uvedený v [20].

4.4.1 Princíp algoritmu

Povedzme, že $A = a_1, a_2, \ldots, a_N$ a $B = b_1, b_2, \ldots, b_M$ sú sekvencie o dĺžke N a M. Editovací graf sekvencií A a B má uzly v každom bode mriežky $(x, y), x \in [0, N], y \in [0, M]$. Uzly sú spojené pomocou horizontálnych, vertikálnych a diagonálnych orientovaných hrán, vytvárajúc acyklický orientovaný graf. Horizontálne hrany spájajú susedné uzly $(x-1, y) \rightarrow (x, y)$, podobne vertikálne hrany spájajú susedné uzly $(x, y-1) \rightarrow (x, y)$. V prípade ak $a_x = b_x$, potom v grafe existuje diagonálna hrana $(x-1, y-1) \rightarrow (x, y)$. Uzol (x, y) je potom nazývaný bod zhody. Príklad takéhoto grafu pre sekvencie A=`abcabba` B=`cbabac` je na obrázku 4.3 [27].



Obrázek 4.3: Príklad editovacieho grafu sekvencií `abcabba` a `cbabac`. Prevzaté a upravené z [27].

Stopa dĺžky L je sekvencia bodov zhody. Pri vytváraní cesty z (0,0) do (N,M) zo stopy sa jednoducho spoja diagonálne hrany za pomoci horizontálnych a vertikálnych hrán. Vzťah medzi cestou a stopu je taktiež demonštrovaný na obrázku 4.3. Podsekvencia hocijakého

reťazca vznikne vymazaním žiadneho alebo ľubovolného počtu znakov z tohto reťazca. Spoločná podsekvencia dvoch reťazcov A a B je podsekvenciou oboch reťazcov. Každá stopa v editovacom grafe A a B je tiež podsekvenciou týchto reťazcov. Presne povedané $a_{x_1}, a_{x_2}, \ldots, a_{x_L} = b_{x_1}, b_{x_2}, \ldots, b_{x_L} \rightarrow (x, y)$ je spoločná podsekvencia A a B len vtedy ak $(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)$ je stopa v editovacom grafe reťazcov A a B. Editovací skript pre A a B je množina príkazov pridaní a zmazaní znakov, ktoré transformujú reťazec A na B. Je dokázateľné, že každá stopa v editovacom grafe A a B korešponduje k nejakému editovaciemu skritpu [27].

Z vyššie uvedených faktov je zrejmé, že spoločné postupnosti, editovacie skripty, stopy a cesty z (0,0) do (N,M) v editovacom grafe sú izomorfné formalizmy. Hrany každej cesty môžu byť interpretované vo forme príslušnej vzájomnej postupnosti alebo editovacieho skriptu. Každá diagonálna hrana cesty končiaca v (x,y) dáva symbol $a_x (= b_x)$ spoločnej postupnosti, každá horizontálna hrana do bodu (x,y) reprezentuje príkaz zmazania znaku a_x a vertikálna hrana vedúca z (x,y) reprezentuje príkaz pridania znaku b_{y+1} za znak a_x. Z toho vyplýva, že počet horizontálnych a vertikálnych hrán cesty udáva dĺžku príslušného editovacieho skriptu [27].

Problém nájdenia najkratšej spoločnej postupnosti je teda ekvivalentný nájdeniu cesty z (0,0) do (N,M) s maximálnym počtom diagonálnych hrán. Ak by sme ohodnotili všetky hrany v editovacom grafe dvoch reťazcov tak, že vertikálne a horizontálne hrany by mali cenu rovnú jednej a diagonálne hrany cenu rovnú nule, je potom problém nájdenia najkratšej spoločnej postupnosti transformovaný na nájdenie cesty z (0,0) do (N,M) s najmenšou cenou v ohodnotenom editovacom grafe [27].

4.4.2 Popis algoritmu

Povedzme, že D-cesta je cesta začínajúca v (0,0) a má presne D nediagonálnych hrán. 0-cesta musí pozostávať čisto z diagonálnych hrán, z čoho vyplýva, že D-cesta musí pozostávať z(D-1)-cesty, jednej nediagonálnej hrany a sekvencie (môže byť aj prázdna) diagonálnych hrán. Ďalej očíslujeme diagonály v editovacom grafe tak, že diagonála k obsahuje (x,y), pričom platí x-y=k. Potom je zrejmé, že diagonály sú očíslované v intervale (-M, N), kde M,N sú dĺžky porovnávaných sekvencií. V [27] je dokázané, že D-cesta vždy končí na diagonále $k \in \{-D, -D+2, \ldots, D-2, D\}$.

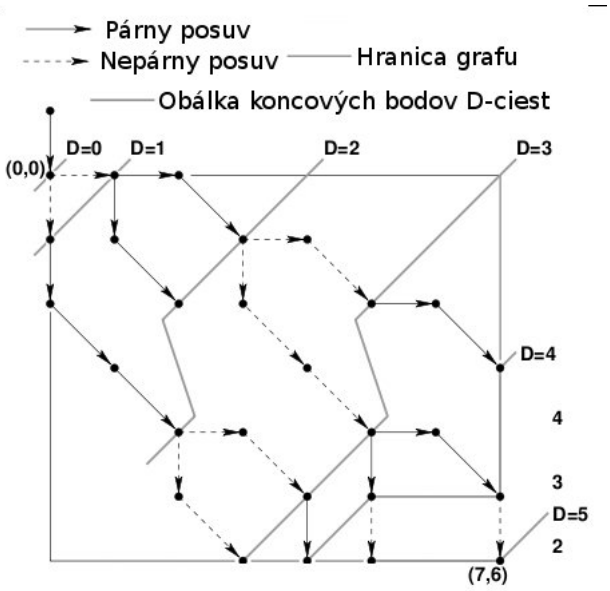
Najdlhšia D-cesta na k diagonále je cesta, ktorá končí na tejto diagonále a zároveň jej koncový bod leží na najväčšom riadku (stĺpci) zo všetkých ciest, končiacich taktiež na tejto diagonále [27].

Ďalej máme koncové body najdlhších (D-1)-ciest na diagonálach $k+1$ a $k-1$, so súradnicami povedzme (x',y'),(x",y"). Potom z týchto ciest získame najdlhšie D-cesty tak, že rozšírime prehľadávanie grafu na diagonálu k tým, že posunieme súradnice na (x',y'+1),(x"+1,y") a následne zídeme po diagonále pokiaľ platí $a_x = b_y$ a zároveň nie je dosiahnutá hranica grafu. Takýmto spôsobom pokračujeme v prehľadávaní grafu až pokiaľ najdlhšia D-cesta na diagonále $N - M$ nekončí v bode (N,M) [27]. Na základe tohoto postupu je možné zostaviť nasledujúci algoritmus uvedený v [27]:

for $D = 0 \rightarrow (M + N)$ **do**
  for $k = D \rightarrow$D s krokom o veľkosti 2 **do**
    Nájdi koncový bod najdlhšej D-cesty na diagonále k.
    if koncový bod = (N,M) **then**
      D-cesta je optimálne riešenie.
      Stop

end if
end for
end for

Na obrázku 4.4 je uvedený príklad prehľadávania editovacieho grafu a spôsobu rozvíjania D-ciest.



Obrázek 4.4: Graf najdlhších D-ciest. Prevzaté a upravené z [27].

4.4.3 Efektívnosť algoritmu

Tento algoritmus pracuje s časovou zložitosťou O((M+N)D), kde M a N sú dĺžky porovnávaných sekvencií a D veľkosť zmeny medzi tými súbormi. Jeho priestorová zložitosť je $O(D^2)$. V [27] sú však uvedené aj varianty tohoto algoritmu, ktoré dosahujú časovú zložitosť $O(M + N + D^2)$ [27].

4.5 Heckelov algoritmus

Algoritmus bol publikovaný v [17], jeho autorom je Paul Heckel a podľa jeho vlastného názoru je jednoduchý a ľahko pochopiteľný, pričom sa vyhýba výpočtovým problémom algoritmov založených na vyhľadaní najdlhšej spoločnej postupnosti. Jeho časová a priestorová zložitosť pre všetky možné prípady je $O(n)$, čiže lineárna [17].

4.5.1 Princíp a popis algoritmu

Tento algoritmus pracuje na základe dvoch predpokladov:

- Riadok, ktorý sa nachádza len jediný raz v oboch súboroch musí byť ten istý riadok, určite nezmenený hoci je možné jeho presunutie. Takto je nájdená väčšina riadkov a sú vylúčené z ďalšieho výpočtu,

- ak sa v každom súbore nachádzajú identické riadky priamo priľahlé k už nájdenému páru riadkov (nájdenému vyššie uvedeným spôsobom), tak to musia byť tie isté riadky. Opakovaná aplikácia tohoto pravidla nájde sekvencie nezmenených riadkov.

Algoritmus pracuje s dvoma súbormi, starým označený O, a novým označený N. Používa tri dátové štruktúry tabuľku symbolov a dve polia OA, NA. Hodnota riadku je používaná ako kľúč do tabuľky symbolov. Každý záznam v tejto tabuľke má dve počítadlá OC a NC, ktoré určujú počet kópií tohoto riadku v príslušných súboroch. Hodnota týchto počítadiel nadobúda hodnoty 0,1 a "veľa". Ďalšou premennou v zázname je OLNO, ktorá udržiava pozíciu riadku v starom súbore a používa sa len v prípade ak je počítadlo OC rovné jednej. Polia OA a NA majú veľkosť rovnú počtu riadkov v príslušných súboroch a záznam v týchto poliach je buď ukazateľ na riadok v tabuľke symbolov alebo pozícia riadku v príslušnom súbore (napríklad číslo riadku a pod.), plus jeden bit na určenie aktuálneho typu hodnoty záznamu (určuje či je záznam ukazateľ alebo pozícia) [17].

Priebeh algoritmu pozostáva zo šiestich jednoduchých priechodov. Pri prvom priechode je v poradí prečítaný každý riadok i súboru N, pričom je vytvorený záznam v tabuľke symbolov (ak ešte neexistuje), je zväčšená hodnota NC počítadla pre tento riadok a hodnota NA[i] je nastavená na ukazateľ do tabuľky symbolov, ktorý ukazuje na záznam riadku i. Druhý priebeh je identický, avšak je čítaný súbor O a sú aktualizované príslušné polia a premenné.

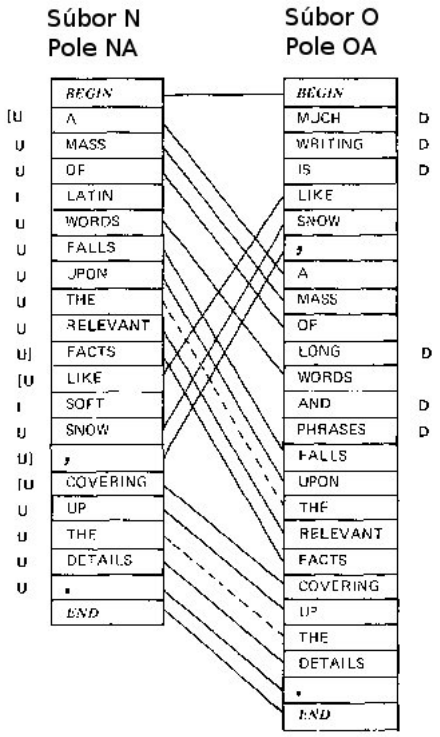
V treťom priechode sa prechádzajú len riadky, pre ktoré platí NC=OC=1. Podľa predpokladu tieto riadky sú tie, ktoré zostali v novom súbore nezmenené. Pre každý takýto riadok v NA a OA vymeníme ukazateľ do tabuľky symbolov za pozíciu riadku v druhom súbore. Napríklad ak NA[i] je takýto riadok, je vyhľadaný v tabuľke symbolov, NA[i] je nastavený na hodnotu OLNO a OA[OLNO] nastavený na hodnotu i. V tomto priechode sú tiež nájdené, resp. pridané unikátne virtuálne riadky nachádzajúce sa pred začiatkom súborov a na po ich konci (niečo ako "začiatok súboru" a "koniec súboru", viď obrázok 4.5) [17].

Pri štvrtom priechode sa prechádzajú všetky riadky vo vzostupnom poradí v poli NA. V prípade, ak NA[i] ukazuje na OA[j] a zároveň NA[i+1] a OA[j+1] obsahujú ukazatele, ktoré ukazujú na rovnaký záznam do tabuľky riadkov, je OA[j+1] nastavené na pozíciu i+1 a NA[i+1] na pozíciu j+1.

V piatom priechode sú riadky v poli NA prechádzané v zostupnom poradí a ak NA[i] ukazuje na OA[j] a zároveň NA[i-1] a OA[j-1] obsahujú ukazatele, ktoré ukazujú na rovnaký záznam do tabuľky riadkov, potom je OA[j-1] nastavené na pozíciu i-1 a NA[i-1] na pozíciu j-1 [17].

V tomto momente NA obsahuje všetky potrebné informácie k tomu, aby boli získane zmeny vykonané na súbore O. V prípade ak NA[i] obsahuje ukazateľ do tabuľky symbolov, potom bol tento riadok vložený a je možné ho označiť ako nový riadok. Ak NA[i] ukazuje na OA[j] a zároveň NA[i+1] neukazuje na OA[j+1], potom je riadok i na hranici zmazaného alebo presunutého bloku a je možné ho príslušne označiť. V poslednom šiestom kroku sú nájdené rozdiely podané na výstup v príslušnom tvare.

Na obrázku 4.5 je možné vidieť výsledné prepojenie polí NA a OA pre dva súbory, obsahujúce anglický text, pričom porovnávané riadky sú v tomto prípade slová. Prvý a posledný záznam v poli sú vyššie spomenuté virtuálne riadky pridané v treťom priechode. Všetky unikátne a zároveň rovnaké riadky v súboroch sú spojené plnou čiarou. Tie sú nájdené tiež v treťom priechode. Rovnaké riadky, avšak nie unikátne, sú spojené čiarkovanou čiarou a sú nájdené v štvrtom a piatom priechode. V poslednom šiestom priechode sú označené zmazané riadky z pôvodného súboru znakom D, riadky pridané do nového súboru znakom I a nezmenené riadky sú označené znakom U [17].

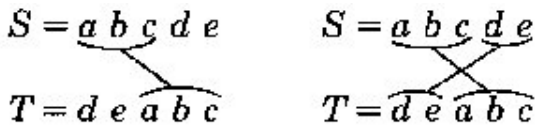


Obrázek 4.5: Výsledné prepojenie polí NA a OA pre súbory A a O. Prevzaté a upravené z [17].

4.6 Tichyho algoritmus

Podľa [36] algoritmy diffu dvoch súborov založené na najdlhšej spoločnej postupnosti týchto súborov nemusia nutne generovať minimálny editovací skript, pretože najdlhšia spoločná postupnosť nemusí obsahovať všetky možné rovnaké sekvencie dvoch porovnávaných súborov.

Algoritmus prezentovaný v [36] je optimálnym algoritmu v takom zmysle, že generuje minimálnu množinu pokrytia podreťazcov jedného reťazca, ktorých jednoduchým presunom je vytvorený druhý porovnávaný reťazec. Napríklad na obrázku 4.6 je vidieť rozdiel medzi možným výstupom niektorého algoritmu založeného na hľadaní najdlhšej spoločnej postupnosti a tohoto algoritmu, kde je v prvom prípade nájdená najdlhšia spoločná postupnosť `abc` a je vytvorený editovací skript, ktorý by vytvoril cieľový reťazec v dvoch krokoch pridaním dvoch znakov "de" pred túto postupnosť a vymazaním znakov `de` za touto postupnosťou. Tento algoritmus vytvorí editovací skript, ktorý by vytvoril cieľový reťazec v jednom kroku, a to presunutím bloku `de` pred blok `abc` [36].



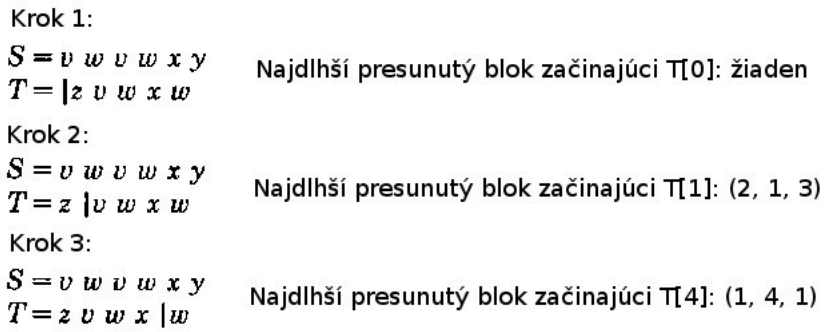
Obrázek 4.6: Ukážka rozdielneho výstupu algoritmu založeného na najdlhšej vzájomnej postupnosti a Tichyho algoritmu. Prevzaté a upravené z [36].

4.6.1 Princíp a popis algoritmu

Algoritmus pracuje s dvoma reťazcami S a T. S je pôvodný reťazec a T je reťazec vytvorený zmenou reťazca S. Algoritmus začína na ľavej strane cieľového reťazca T, a snaží sa nájsť prefixy reťazca T v pôvodnom reťazci S. Toto je dosiahnuté jednoducho prehľadávaním reťazca S zľava doprava a hľadaním najdlhšieho prefixu.

Jednu iteráciu hľadania prefixov je možné ukončiť v momente keď v reťazci S zostane neprehľadaných menej ako $l+1$ znakov, kde l je dĺžka najdlhšieho prefixu nájdeného v tejto iterácii. Ak nie je nájdený žiaden prefix reťazca T, odstráni sa z T prvý symbol a znovu sa hľadajú prefixy. V prípade ak sú nájdené nejaké prefixy, vyberie sa ten najdlhší a uloží sa ako presunutý blok. Následne sa tento prefix vymaže z reťazca T a znovu sa začne hľadať prefix zostávajúceho reťazca T od začiatku S.

Tento postup sa opakuje kým nie je reťazec T prázdny, čiže sú vymazané všetky symboly z tohoto reťazca. Hľadanie prefixov je tiež možné ukončiť v prípade ak niektorý nájdený presunutý blok (prefix) obsahuje T[n], kde n je dĺžka reťazca T. Uložené presunuté bloky tvoria minimálnu množinu pokrytia presunutých blokov. Na obrázku 4.7 je znázornených niekoľko prvých krokov algoritmu pre porovnanie reťazcov `vwvwxy` a `zvwxw`.



Obrázek 4.7: Nájdené presunuté bloky pre reťazce `vwvwxy` a `zvwxw`. Prevzaté a upravené z [36].

Na obrázku 4.7 je najdlhší nájdený prefix uvedený za dvojbodkou a prvá hodnota v zátvorkách je pozícia prefixu v pôvodnom reťazci S, druhá hodnota pozícia v reťazci T a tretia hodnota je dĺžka nájdeného prefixu. V prvom kroku je hľadaný neprázdny prefix reťazca T[0, ..., 4] v reťazci S[0, ..., 5]. Žiadny prefix však nebol nájdený, takže je odstránený prvý symbol z reťazca T a v druhom kroku je hľadaný prefix reťazca T[1, ..., 4]. V tomto prípade boli nájdené dva prefixy s rôznou dĺžkou a je vybraný ten dlhší s dĺžkou troch symbolov začínajúci v pôvodnom reťazci na druhej pozícii, pričom symboly sú číslované zľava, prvý na pozícii nula. V treťom kroku je hľadaný T[5] v reťazci S[0, ..., 5] a sú nájdené znovu dva prefixy, avšak tento krát s rovnakou dĺžkou. Je možné zvoliť ľubovoľný z nich. V tomto momente je reťazec T prázdny a algoritmus nájdenia minimálnej množiny pokrytia presunutých blokov skončí [36].

Dôkaz, že tento algoritmus nájde minimálnu množinu pokrytia presunutých blokov je v [36]. Avšak problémom je jeho veľká časová zložitosť. Preto sú v [36] uvedené vylepšenia, ktoré dokážu tento problém vyriešiť.

Prvé vylepšenie je vhodné v prípade ak pôvodný reťazec S pozostáva len z množiny veľmi málo opakovaných symbolov. Povedzme, že α je počet symbolov abecedy v S a zároveň α je približne rovné m, čo je dĺžka reťazca S. V tomto prípade je možné použiť index, ktorý pre každý symbol s abecedy obsahuje zoznam všetkých pozícií s v reťazci S. V tomto prípade je

možné zmeniť vyhľadávanie prefixov nasledovne.

- T[q]=s je prvý symbol v dosiaľ neporovnanej časti reťazca T,
- algoritmus vyhľadá pomocou indexu zoznam všetkých pozícií tohoto symbolu v S,
- ak je zoznam prázdny, prefix začínajúci symbolom s sa nenachádza v reťazci S a je možné skončiť aktuálne vyhľadávanie prefixov začínajúcich týmto symbolom
- inak je nájdený najdlhší presunutý blok (prefix) medzi tými, ktoré začínajú symbolom s v reťazci S.

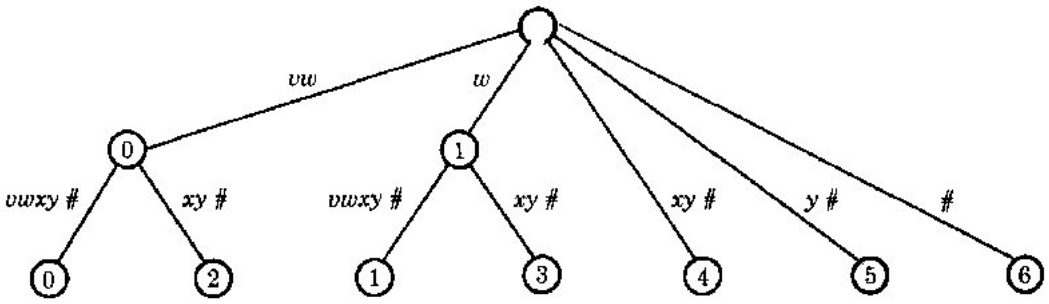
Pre odhad časovej zložitosti takto upraveného algoritmu predpokladajme, že priemerná dĺžka presunutých blokov je l. V tomto prípade každý maximálny presunutý blok bol vybratý spomedzi m/α možností za cenu nie viac ako l+1 porovnaní pre každý blok. Potom časová zložitosť algoritmu je $O(l(m/\alpha)(n/l\ ))= O(mn/\alpha)$, čo je skoro lineárna zložitosť v prípade ak m je približne rovné α. Samozrejme je nutné vytvoriť index v lineárnom čase, čo je možné buď použitím hašovania alebo vytvorením tabuľky o veľkosti abecedy. Vlastnosti potrebné pre toto vylepšenie majú napríklad texty programov a súvislý text prózy, namiesto znakov sa však porovnávajú celé riadky, pričom ďalšie zrýchlenie je možné získať použitím hašovania riadkov.

Druhá možnosť ako dosiahnuť lineárnu časovú zložitosť je použiť v tomto algoritme pokročilé dátové štruktúry. Keďže základný algoritmus pri hľadaní prefixov reťazca T opakovane prehľadáva pôvodný reťazec S, je vhodné predspracovanie tohoto reťazca vedúce k urýchleniu algoritmu. Vhodnou dátovou štruktúrou je v tomto prípade strom sufixov. Najprv sa k pôvodnému reťazcu S pridá na koniec unikátny symbol # a následne sa z takto vzniknutého reťazca S# vytvorí strom sufixov. Pri takomto predspracovaní pôvodného reťazca S je možné nájsť prefix v čase závislom od dĺžky tohoto prefixu. Keďže takýto strom sufixov je možné zostrojiť v časovej a priestorovej zložitosti závislej od dĺžky reťazca, má takto upravený algoritmus zložitosť O(m+n).

Každá hrana vo vytvorenom strome sufixov je asociovaná s nejakým podreťazcom reťazca S. Pre každú cestu z koreňa do listu tohoto stromu, tieto asociované podreťazce vytvoria sufix reťazca S. Počiatočný bod sufixu reprezentovaného cestou v strome sufixov v originálnom reťazci S je uložený v listovom uzly tejto cesty v strome sufixov. Sufixy so spoločným prefixom majú rovnaký rodičovský uzol a cesta z koreňa do tohoto uzlu reprezentuje tento spoločný prefix. V tomto vnútornom uzly je uložený počiatočný bod jedného (ľubovoľného) sufixu v pôvodnom reťazci S. Na obrázku 4.8 môžeme vidieť strom sufixov pre reťazec `vwvwxy` a vnútorný uzol obsahuje pozíciu najľavejšieho sufixu.

Pri hľadaní spoločného prefixu P reťazcov S a T je vo všeobecnosti možné ukončiť toto prehľadávanie z troch dôvodov:

- Je nájdená cesta, ktorá reprezentuje sufix rovnaký ako prefix P,
- je dosiahnutý niektorý z listových uzlov tohoto stromu, avšak nie je dosiahnutý aj posledný znak v hľadanom prefixe,
- žiadna hrana v strome sufixov vedúca z aktuálne navštíveného interného uzlu neumožňuje ďalšie valídne prehľadávanie stromu, tj. porovnávaná časť hľadaného prefixu nie je rovnaká ako niektorý podreťazec asociovaný s hranami vedúcimi z aktuálne navštíveného interného uzlu



Obrázek 4.8: Strom sufixov pre reťazec `vwvwxy`. Obrázok prevzatý z [36].

Ak dôjde k ukončeniu hľadania prefixu z posledných dvoch spomenutých dôvodov, takýto prefix sa v pôvodnom reťazci S nenachádza. Cesta v strome sufixov nájde vždy najdlhší možný prefix P v čase závislom od jeho dĺžky. Ďalej pre dosiahnutie efektívneho vyhľadávania v sufixovom strome je vhodné do jeho hrán zakódovať indexy do hašovacej tabuľky namiesto hodnôt podreťazcov reťazca S.

Pre vyššie uvedený príklad diffu reťazcov S = vwvwxy a T = zvwxw by algoritmus po takomto vylepšení prebiehal nasledovne:

1. Hľadanie prefixu T[0, ..., 4] v sufixovom strome skončí pri koreni stromu, pretože žiadna hrana označená ako `z` z koreňa nevedie - v S sa nenachádza žiaden prefix T[0, ..., 4],

2. hľadá sa prefix T[1, ..., 4]. Toto hľadanie skončí pri listovom uzly s uloženou hodnotou dva - je nájdený presunutý blok `(2,1,3)`,

3. pri poslednom hľadaní prefixu T[4] sa prehľadávanie ukončí na vnútornom uzly - je možné použiť presunutý blok (1,4,1) alebo blok `(3,4,1)`.

Editačný skript, ktorý transformuje pôvodný reťazec S na reťazec T je sekvencia príkazov `pridaj` a `presuň`. Tieto príkazy vytvoria reťazec T zľava doprava. Každý príkaz presunutia bloku `(p, q, l)` je vo forme `M p l`, ktorý skopíruje reťazec S[p, ..., p+l-1] na koniec reťazca T. Pre doplnenie podreťazcov T[u, , v] reťazca T, ktoré pozostávajú zo znakov nenachádzajúcich sa v reťazci S, obsahuje editačný skript príkaz `A T[u, ..., v]`, ktorý pridá tento podreťazec na koniec aktuálne vytváraného T.

Vo všeobecnosti nie je možné vytvoriť reťazec T jedným priechodom, pretože presunuté bloky sa môžu križiť. To môže byť problém pri práci s veľkými súbormi alebo inými štruktúrami so sekvenčným prístupom. V takomto prípade je možné algoritmus optimalizovať vybratím vhodnejšieho presunutého bloku, v prípade ak sa tento presunutý blok nachádza na viacerých miestach v S. Takýmto spôsobom sa zníži počet navracaní v sekvenčnej štruktúre.

Napríklad ak jeden presunutý blok má začiatok v reťazci S na pozíciách p a q, p<q a predchádzajúci nájdený presunutý blok má posledný znak niekde medzi pozíciami p a q je vhodnejšie použiť presunutý blok so začiatkom v q. V prípade ak je v algoritme použitá prvé zmienené vylepšenie je cena hľadania všetkých pozícií presunutého bloku zanedbateľná. Pri použití druhého vylepšenia by však prehľadávanie všetkých pozícií degradovalo časovú zložitosť na O(mn), takže by bolo nutné vytvoriť ďalšiu dátovú štruktúru pre vnútorné uzly, aby bola časová zložitosť zlepšená.

Ďalšou možnosťou je vytvorenie reťazca T priamo na mieste pôvodného reťazca S, čo môže byť užitočné najmä v prípadoch ak na pamäťovom médiu nie je pre ďalší reťazec T miesto (treba pamätať na to, že miesto reťazca môžeme uvažovať napríklad video záznam a znaky by reprezentovali sekvencie tohoto záznamu) [36].

4.6.2 Efektívnosť algoritmu

Základný algoritmus pracuje s časovou zložitosťou O(mn) a priestorovou zložitosťou O(m+n). Prvý vylepšený variant má v takmer lineárnu zložitosť O(n) a priestorová zložitosť je závislá od použitého spôsobu indexovania. Druhý variant dosahuje priamo lineárnu zložitosť O(n) a priestorovú zložitosť O(m+n) [36].

Kapitola 5

Technológie

Keďže úlohou tejto práce nie je vyvíjať nové nástroje pre extrakciu charakteristických rysov z audia, prípadne iných nástrojov pre prácu so zvukom, budú v prípade potreby použité už existujúce implementácie takýchto typov aplikácií. V tejto kapitole sú uvedené informácie o nástrojoch, ktoré budú slúžiť pri rozhodovaní, ktorý nástroj je vhodnejšie použiť pri implementácii aplikácie diffu audio dokumentov. V prvej časti sú popísané nástroje pre extrakciu znakov z audia, po nich nasledujú aplikácie poskytujúce API pre prácu so zvukom a v poslednej časti sú v krátkosti predstavené programovacie jazyky použité pri implementácii.

5.1 HTK a Kaldi

HTK (Hidden Markov Model Toolkit) sú prenosné nástroje pre vytváranie a manipuláciu so skrytými Markovovmi modelmi. HTK je primárne používané v oblasti rozpoznávania reči, avšak našlo tiež využitie v ďalších oblastiach ako napríklad vo výskume syntézy reči či sekvencovania DNA. HTK pozostáva z modulov a nástrojov dostupných vo forme zdrojových kódov jazyka C. Tieto nástroje umožňujú analýzu reči, trénovanie skrytých Markovových modelov, ich testovanie a následnú analýzu [4].

Prvá verzia HTK bola vyvinutá na oddelení reči, videnia a robotiky (Speech Vision and Robotics Group) na strojárskej fakulte Univerzity Cambridge v roku 1989. Jej autorom bol Steve Young a bola používaná v rámci tohoto oddelenia vo výskume rozpoznávania reči. Časom sa počet vývojárov rozrástol. V roku 1993 prevzali vývoj, správu a aj práva k HTK laboratória Entropic Research (ERL) a v roku 1995 vznikli spojením ERL a Univerzity Cambridge laboratóriá Entropic Cambridge Research (ECRL). V novembri roku 1999 boli ECRL kúpené spoločnosťou Microsoft, ktorá takto získala práva k HTK avšak umožnila ďalší voľný vývoj a distribúciu tým, že licenciu HTK vrátila späť na CUED a od roku 2000 umožnila stiahnutie zdrojových kódov HTK z domovských stránok CUED [12].

Podobne ako HTK, Kaldi sú nástroje pre vytváranie a trénovanie skrytých Markovových modelov používaných v oblasti rozpoznávania reči. Tieto nástroje sú voľne šíriteľné a editovateľné pod GPL licenciou. Ich vývoj sa v spolupráci s mnohými odborníkmi uskutočnil na Fakulte informačných technológií Technickej univerzity v Brne. Táto knižnica je taktiež implementovaná v jazyku C/C++ [3].

5.2 Nástroje pre prácu so zvukom

Vyššie uvedené nástroje, ktoré budú použité pre extrakciu charakteristických znakov z porovnávaného audia dokážu pracovať len s určitými typmi audio súborov a keďže v súčastnosti existuje veľa používaných formátov audio súborov, je nutné nejakým spôsobom vytvoriť z audio súboru nepodporovaného nástrojmi Kaldi alebo HTK audio súbor, ktorý je nimi podporovaný. Túto funkciu zabezpečujú nástroje pre konvertovanie audia ako napríklad FFmpeg alebo GStreamer.

FFmpeg je mult-platformové riešenie pre nahrávanie, konvertovanie a streamovanie audia a videa s intuitívnym riadkovým rozhraním, čo znamená, že FFmpeg sa snaží automaticky vyhodnotiť všetky parametre a jediný parameter, ktorý musí zadať užívateľ je cieľovú bitovú rýchlosť (bitrate). Tento nástroj je šíriteľný pod licenciou LGPL alebo GPL [1].

Gstreamer je rozhranie pre tvorbu aplikácií streamovania rôznych médií (audio, video). Vývoj základného dizajnu Gstreameru bol inšpirovaný video rúrou na Oregon Graduate Institute a aplikáciou DirectShow. Bol navrhnutý tak, aby umožnil jednoduché vytváranie aplikácií pracujúcich z audiom a videom a podporuje širokú škálu formátov. Podporuje nielen audio a video, ale dokáže spracovať hocijaký prúd dát. Je založený na systéme pluginov, ktoré poskytujú rôzne kodeky a inú funkcionalitu. Tieto pluginy môžu byť zreťazené do rúry a takto definovať dátový prúd. Tieto rúry môžu byť editované pomocou grafického editoru a poprípade uložené do formátu XML. GStreamer používa objektový model GObject, ktorý je použitý v GLib 2.0. GStreamer je implementovaný takým spôsobom, aby bola programovacia metodika podobná GTK+. Toto sa vzťahuje na objektový model, vlastníctvo objektov, referencovania a podobne. Je možné si stiahnuť zdrojové kódy tohoto nástroja online s LGPL licenciou [35].

5.3 Použité programovacie jazyky

Ako už bolo uvedené aplikácia, ktorej vývoj je predmetom tejto práce bude implementovaná v dvoch programovacích jazykoch, a to v jazyku Python a jazyku C++.

Python je moderný objektovo orientovaný otvorený (open source) skriptovací jazyk, ktorý bol vyvinutý Guido van Rossumem. Tento jazyk nie je závislý na platforme, čo znamená, že je možné ho použiť na rôznych operačných systémoch. Python je zameraný na rýchly a jednoduchý vývoj aplikácií, operácie sú vykonávané na vyššej úrovni abstrakcie (napríklad v porovnaní s jazykmi ako C,C++), je charakteristický jednoduchými pravidlami syntaxe. Toto si však vyberá daň v podobe pomalšieho behu vytvorených aplikácií v porovnaní s jazykmi ako C,C++, najmä pri výpočtovo náročných operáciách. Avšak Python poskytuje elegantné riešenie tým, že umožňuje skombinovať kód písaný v jazyku Python s modulmi napísanými v iných jazykoch, a tak je možné vyvíjať väčšinu programu v jazyku Python a výpočtovo náročné časti písať napríklad v moduloch jazyka C alebo C+ +. Python ďalej poskytuje automatickú správu pamäte, jednoduchý prístup k nástrojom pre tvorbu grafických užívateľských rozhraní, zabudované mechanizmami pre perzistentné ukladanie objektov, vyspelé asociatívne polia a mnoho ďalších funkcií a knižníc uľahčujúcich programátorovi život [15].

Vývoj jazyka C++ sa začal Bjarnom Stroustrupom v roku 1980. Tento jazyk vznikol vylepšením, pozmenením a pridaním veľa nových vlastností už existujúcemu jazyku C. Medzi najdôležitejšie vlastnosti patrí zavedenie objektového prístupu k programovaniu. Programy implementované v tomto jazyku sú efektívnejšie a rýchlejšie ako implementácie vo väčšine iných programovacích jazykoch [29].

Kapitola 6

Návrh a implementácia aplikácie

V predchádzajúcich kapitolách boli uvedené informácie, ktoré slúžili ako teoretický základ pre návrh a implementáciu tejto aplikácie. V tejto kapitole bude v prvej časti podrobne popísaný jej návrh a dôvody, ktoré k takémuto návrhu viedli. V druhej časti je popísaná implementácia, použité algoritmy a ich vplyv na efektivitu cieľovej aplikácie a správnosť jej výstupu.

6.1 Návrh programu

Ako bolo spomenuté vyššie, úlohou algoritmu diffu je nájdenie vzájomných rozdielov medzi dvoma súbormi. V našom prípade ide o audio súbory a úlohou tejto aplikácie je nájdenie vzájomných ľudským uchom počuteľných rozdielov v týchto súboroch. Aby bolo možné splniť toto kritérium, nie je vhodné porovnávať tieto súbory po vzorkách uložených v audio súbore, pretože takýmto spôsobom nie je možné získať požadované informácie o zaznamenanom zvuku v širšom kontexte. Z tohoto dôvodu je nutné nejakým spôsobom získať takú reprezentáciu zvukového signálu, aby dokázala aproximovať, čo v skutočnosti počuje ľudské ucho.

Ďalším problémom je, že všetky algoritmy diffu pracujú so sekvenciami nejakých atomických hodnôt, riadkov, prípadne znakov. Pri práci so spojitým signálom (hoci je v skutočnosti diskretizovaný pomocou PCA) je však zrejmé, že neexistuje žiadne prirodzené rozdelenie tohoto signálu na atomické hodnoty (keďže použitie priamo uložených vzoriek nie je vhodné). V oblasti rozpoznávania reči tento problém obchádzajú napríklad tak, že signál rozdelia na slová, pričom slovo je nájdené pomocou algoritmov sledujúcich energiu a apmlitúdu signálu. Tento prístup však nie je možné použiť, pretože je nutné, aby cieľová aplikácia pracovala so všetkými typmi zvuku.

Keďže pri analýze a spracovaní spojitého signálu sa vždy tento signál rozdelí na malé úseky použitím okna v procese windowing, bolo rozhodnuté, že sa signál rozdelí týmto spôsobom a časť signálu ohraničená jedným oknom bude slúžiť ako atomická jednotka v samotnom algoritme diffu (analógia k riadkom). Všetky problémy, ktoré takýmto rozdelením signálu vzniknú, bude nutné riešiť inými technikami. Medzi problémy tohoto prístupu patrí napríklad zlé zarovnanie okien medzi dvoma audio súbormi (viď nižšie) alebo v prípade dlhšieho okna sa môže stať, že v okne bude ležať hranica zmeny audio signálu (napríklad zmena hudby na ticho), čím sa táto hranica rozmaže.

Z každej časti zvukovej vlny ohraničenej oknom sa následne extrahuje vektor charakteristických znakov, ktorý by mali poskytnúť širšiu informáciu o tom, čo v skutočnosti zazna-

mená sluchový aparát pri jej dopade na ušný bubienok. Pre tento účel budú použité funkcie knižnice pre extrakciu charakteristických znakov zvuku. Na základe dostupných informácií bolo rozhodnuté, že sa budú extrahovať koeficienty cepstra mel-frekvencie. Dôvodom je veľká podpora takejto reprezentácie zvuku a navyše aj široká doména problémov, ktoré boli riešené s ich pomocou.

Na takýmto spôsobom získané dáta z dvoch porovnávaných audio signálov, sa potom použije implementovaný algoritmus diffu, ktorý nájde ich vzájomné rozdiely a označí každú atomickú jednotku v oboch sekvenciách ako zmenenú alebo nezmenenú. Na základe tohoto výstupu je možné generovať požadovaný editovací skript alebo inú formu zobrazenia vzájomných rozdielov dvoch sekvencií. Formálne je možné tento systém popísať následovne:

- Zvukový signál je usporiadaná n-tica celých čísiel $S = (i_1, i_2, \ldots, i_N)$, $i_1, i_2, \ldots, i_N \in Z$,
- proces extrakcie charakteristických znakov signálu je funkcia f_e, ktorá zobrazuje časť n-tice s do usporiadanej n-tice reálnych čísiel $f_e : (i_j, i_{j+1}, \ldots, i_k) \to E$, $(i_j, i_{j+1}, \ldots, i_k) \in S$, $E = (e_1, e_2, \ldots, e_m)$, $e_1, e_2, \ldots, e_m \in R$, $m \ll k$,
- po procese extrakcie znakov je audio signál reprezentovaný usporiadanou n-ticou n-tíc $S_e = (e_1, e_2, \ldots, e_m)$, $e_1, e_2, \ldots, e_m \in E$,
- definujme C ako usporiadanú n-ticu $C = (c_1, c_2, \ldots, c_m)$, $\forall c_i \in \{True_i, False_i\}$, $1 \leq i \leq m$, kde i je pozícia c v n-tici, potom $True_i$ znamená, že i-tý prvok bol zmenený v sekvencii, analogicky $False_i$ znamená, že i-tý prvok nebol zmenený.

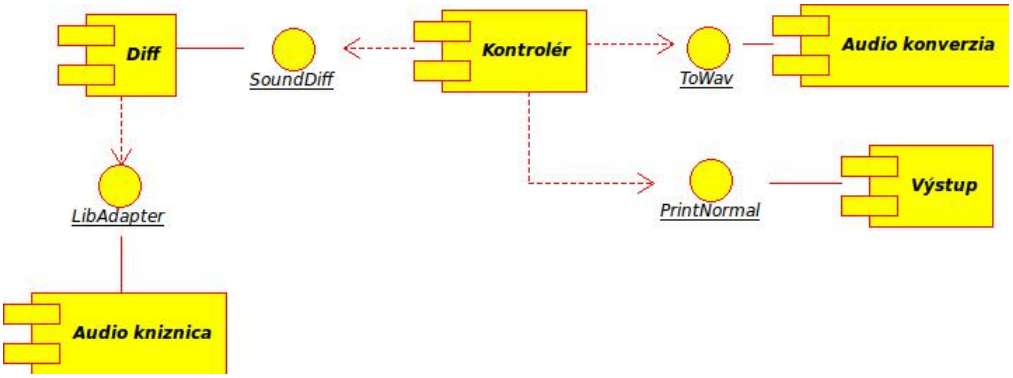
  Algoritmus diff je funkcia f_d, ktorá zobrazuje dve usporiadané n-tice n-tíc na usporiadanú dvojicu n-tíc $f_d : (s_1, s_2) \to (o_1, o_2)$, $s_1, s_2 \in S_e$ $o_1, o_2 \in C$,
- zobrazenie výstupu je funkcia f_o, ktorá zobrazí usporiadanú dvojicu n-tíc na ľudsky pochopiteľný výstup $f_o : (i_1, i_2) \to OUT$, $i_1, i_2 \in C$, OUT je požadovaný výstup.

Po formálnej špecifikácii problému bolo pristúpené k návrhu celej aplikácie. Počas tohoto procesu boli brané do úvahy nielen funkčné požiadavky na cieľovú aplikáciu, ale aj rôzne obmedzenia, vyplývajúce z použitia rozličných programovacích jazykov a softwarových nástrojov. Keďže bol navyše kladený veľký dôraz na efektivitu tejto aplikácie, museli v určitých prípadoch ustúpiť zásady dobrého návrhu v prospech získania nižšej výpočtovej náročnosti. Konečný návrh aplikácie je možné vidieť na obrázku 6.1, kde je zobrazený graf komponentov tejto aplikácie.

Táto aplikácia sa skladá zo piatich komponentov. Ústredným komponentom je Kontrolér, ktorý riadi činnosť celej aplikácie. Z obrázku 6.1 je zrejmé, že Kontrolér neovláda extrakciu znakov audio signálu a táto časť je ovládaná priamo z komponentu Diff.

Aplikácia bola takto navrhnutá z toho dôvodu, že všetky dostupné knižnice pre extrakciu charakteristických znakov audia boli vytvorené v jazyku C/C++ a taktiež bolo rozhodnuté, že algoritmus diffu bude kvôli jeho výpočtovej náročnosti implementovaný v jazyku C/C++. Toto je dôvod prečo Kontrolér nemá možnosť ovládať extrakciu znakov z audia, hoci pri zlyhaní tohoto procesu nie je možné vykonať proces diff, a preto by nemali byť vôbec zavolané funkcie komponenty Diff.

Pokiaľ by však bolo umožnené Kontroléru priamo získavať dáta z komponentu Audio knižnica a predávať ich komponentu Diff (v prípade ak nenastane žiadna chyba), bolo by nutné dáta získané v komponente Audio knižnica konvertovať z reprezentácie dát, s ktorou



Obrázek 6.1: Graf komponentov vytvorenej aplikácie.

sa pracuje v jazyku C/C++, do dát vhodných pre python a predať ich Kontroléru. Pri ich následnom predávaní z Kontroléru do komponentu Diff, by sa znova musela vykonať opačná konverzia. Tým, že komponent Diff priamo riadi extrahovanie charakteristických znakov audia, bolo umožnené vyhnúť sa týmto operáciám a ušetriť výpočtové nároky. Na druhej strane pokiaľ nastane nejaká chyba v procese extrakcie, musí túto chybu spracovať komponent Diff a posunúť ju ďalej Kontroléru, aby mohli byť vykonané patričné opatrenia a užívateľovi bola oznámená vzniknutá chyba.

6.2 Implementácia a použité algoritmy

Ako bolo vyššie uvedené ústredným komponentom je Kontrolér. Tento komponent je implementovaný v programovacom jazyku python a je reprezentovaný modulom `Main.py`. V tomto module sa spracúvajú parametre uvedené pri spustení aplikácie, kontroluje je sa ich správnosť a úplnosť a spúšťa sa samotný proces diffu so zadanými parametrami. Je tiež zodpovedný za správnu reakciu v prípade vzniknutia chyby v module obsiahnutom v niektorom z jeho podriadených komponentov.

Keďže extrakcia charakteristických znakov audio signálu pomocou knižníc uvedených v kapitole 5.1 nie je možná z akéhokoľvek formátu audio súboru, je nutné formáty, s ktorými tieto knižnice pracovať nevedia, prekódovať do zodpovedajúceho formátu. Za túto činnosť je zodpovedný komponent Audio konverzia, ktorý bol implementovaný v programovacom jazyku python. Tento komponent reprezentovaný triedu `ToWav` nachádzajúcou sa v rovnomennom module. Táto trieda poskytuje funkcie pre konverziu audio súborov. Vstupný údajom je cesta k zdrojovému audio súboru a cesta kde má byť vytvorená jeho kópia zakódovaná do cieľového formátu. Vždy je overené, či zdrojový súbor vôbec existuje. Ak nie, je vyhodená výnimka, ktorá je zachytená v hlavnom module a na chybový výstup je vypísaná chybová hláška. V opačnom prípade je z tohto audio súboru vytvorený dočasný súbor vo formáte wav, ktorý je následne uložený do cieľovej zložky. Východzím nastavením je zložka /tmp/.sounddiff. Taktiež sú z tohto súboru odstránené všetky meta dáta, pretože vybraná knižnica (viď kapitolu 7.1) nedokáže spracovať takéto audio. Pre konverziu audio súborov je využívaný nástroj gstreamer. Konkrétne nástroje gstreameru spustiteľné z príkazovej riadky. K takémuto riešeniu som sa uchýlil z toho dôvodu, že pre python verzie tri, zatiaľ neexistuje podpora gstreamer API.

Posledným komponentom vytvoreným v jazyku python je komponent Výstup, ktorý je reprezentovaný triedou `printNormal` v module `printNormal.py`. Táto trieda na základe

výstupu z komponentu Diff zostaví ľudsky čitateľný výstup a vypíše ho na obrazovku. Tento výstup bol vytvorený pozmenením normálneho výstupu klasického unixového nástroja diff tak, že miesto čísiel riadkov sú zobrazené časové intervaly jednotlivých signálov. Pre užívateľa, ktorý pozná unixový diff, bude výstup z cieľovej aplikácie ľahko pochopiteľný. Jednotky času sú udávané v sekundách. Napríklad:

- `0.00,0.40c0.00,0.40` znamená zmeniť interval 0s–0.4s v prvom súbore za 0s–0.4s v druhom súbore,
- `59.90,60.00d59.90` znamená zmazať interval 59.90s–60.00s v prvom súbore,
- `30.40a40.40,50.50` znamená za prvých 30,4s v prvom súbore pridať interval 40.40s–50.50s z druhého súboru.
- pokiaľ sa v nahrávke nachádza viac kanálov hudby (nejedná sa o mono nahrávku), sú jednotlivé kanály porovnané tak, že prvý kanál je porovnaný s prvím, druhý s druhým atď. Výstupy pre jednotlivé kanály sú oddelené znakom `--------`

To znamená, že výstup porovnania dvoch audio súborov môže vyzerať napríklad takto:

```
0.00,0.20c0.00,10.20
15.00a25.00,35.10
30.40,35.40d50.40
59.90a99.90,101.00
---------------
0.00,0.70c0.00,10.00
17.00a20.20,31.00
```

Ďalším a azda najdôležitejším komponentom je Audio knižnica. Táto časť sa stará o extrakciu charakteristických znakov audio signálu a jeho výstup je vstupom pre samotný algoritmus diffu. Z tohoto dôvodu je veľmi dôležité získať čo najlepšiu reprezentáciu signálu. To znamená, aby dva signály s rôznym zvukom boli reprezentované čo možno najodlišnejšími hodnotami. V opačnom prípade by algoritmus diffu pracoval s nesprávnymi dátami, vedúc tak k nesprávnym výsledkom.

Tento komponent pozostáva z funkcií, ktoré poskytuje vybraná knižnica pre extrakciu charakteristických znakov zvuku. Keďže som sa snažil, aby bola prípadná výmena tejto knižnice (prípadne zmena jej verzie) rovnako ako kompilácia celej aplikácie jednoduchá, je táto časť spojená s ostatnými časťami len typom dát, ktoré poskytuje. To znamená, že som sa snažil o to, aby do maximálnej možnej miery boli vnútorné štruktúry knižnice skryté pred zvyškom aplikácie. Z tohoto dôvodu som do zdrojových kódov knižnice pridal tri ďalšie triedy. Trieda `ExtractMfcc` sa stará o samotnú extrakciu charakteristických dát zo zadaného audio súboru, pričom používa funkcie a vnútorné štruktúry knižnice k extrakcii a uloženiu získaných dát. Preto som vytvoril ďalšiu triedu nazvanú `LibAdapter`, ktorá oddeľuje knižnicu od zvyšku aplikácie. Táto trieda využíva služby triedy `ExtractMfcc` a poskytuje dáta vo forme objektov triedy `sItem`. Taktiež bola vytvorená štruktúra `Settings`, ktorou je možné zmeniť východzie nastavenia parametrov extrakcie znakov audio signálu, umožňujúc tak väčšiu variabilitu celej aplikácie. Trieda `sItem` je potomkom triedy `Item` (viď nižšie) a slúži ako reprezentácia úseku signálu vymedzeného veľkosťou jedného okna v procese windowing, čiže predstavuje spomínanú atomickú hodnotu. Trieda tiež implementuje vzájomné porovnanie dvoch objektov tejto triedy, pomocou preťaženia operátoru porovnania `==`. Nastavenia pri extrakcii dát sú uvedené v kapitole 7.2.

Na základe výsledkov experimentov uvedených v kapitole 7.2, bolo rozhodnuté, že každý objekt triedy `sItem` obsahuje dva vektory koeficientov (reálnych čísiel) reprezentujúcich príslušnú časť audio signálu. Jeden vektor pozostáva z dvanástich koeficientov cepstra mel-frekvencie vypočítaných z tejto časti signálu. Druhý obsahuje tri koeficienty získané pomocou algoritmu SVD pri redukcii dimenzionality matice extrahovaných dát. To znamená, že pokiaľ pri tejto redukcii vzniká k-aproximácia matice dát, potom o druhom vektore koeficientov môžeme povedať, že je k-aproximáciou prvého vektoru (v našom prípade k=3). Alebo inak, bod definovaný súradnicami v tomto vektore je k-aproximáciou bodu určeného priamo MFC koeficientami.

Takže každý úsek signálu je reprezentovaný dvoma rôznymi bodmi v n-rozmernom priestore, kde n je rovné dvanástim v prípade prvého vektoru súradníc, a trom v prípade vektoru druhého. Toto je východisková pozícia pre porovnávanie podobnosti dvoch objektov triedy `sItem`. Keďže je samozrejmá určitá variácia vypočítaných MFC koeficientov pre (skoro) rovnaké signály, nie je možné jednoduché porovnávanie na presnú zhodu ako riadky v textovom súbore.

Z toho dôvodu, rozhodnutie, či sú dva body rovnaké alebo nie, závisí od ich vzájomnej vzdialenosti. V implementácii je počítaná ich euklidovská vzdialenosť, avšak bez použitia odmocnenia, čiže len ako súčet druhých mocnín rozdielov koeficientov, ktoré jednotlivé objekty obsahujú. Vynechanie odmocnenia ušetrí veľkú časť výpočtových nákladov, čo vedie k zrýchleniu algoritmu. Formálny zápis implementovanej vzdialenosti dvoch bodov je nasledovný:

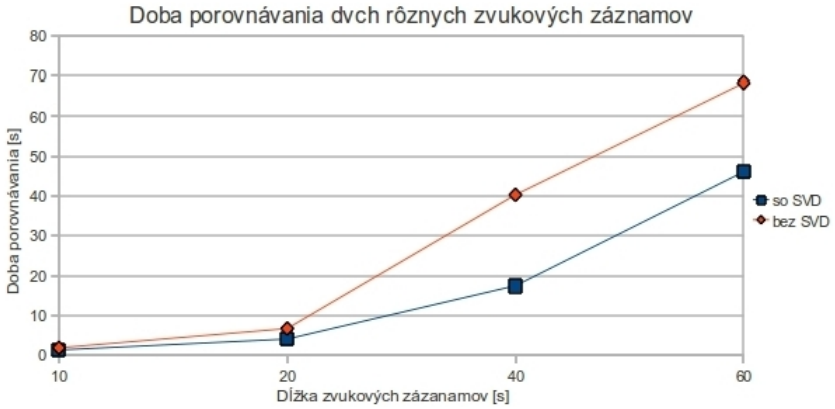
$$DIST((i_1, i_2, \ldots, i_n), (j_1, j_2, \ldots, j_n)) = \sum_{k=1}^{n} (i_k - j_k)^2 \tag{6.1}$$

V samotnom procese porovnania sa potom takýmto spôsobom najprv vypočíta vzdialenosť bodov určených troma súradnicami (vektor získaný pomocou SVD). Pokiaľ je táto vzdialenosť väčšia ako hraničná hodnota, sú porovnávané objekty vyhodnotené ako rôzne. V opačnom prípade sa porovná vzdialenosť bodov určených súradnicami uloženými v druhom vektore a ak je menšia resp. väčšia ako druhá hraničná hodnota, sú tieto objekty vyhodnotené ako rovnaké resp. odlišné. Veľkosť druhej hraničnej hodnoty H_2 použitej v druhom porovnaní, bola na základe výsledkov experimentov v kapitole 7.2 nastavená na 1300. Veľkosť hraničnej hodnoty H_1 použitej v prvom porovnávaní je kvôli problémom s algoritmom SVD (viď 7.2.4) premenlivá. Tento postup ilustruje nasledujúci pseudokód:

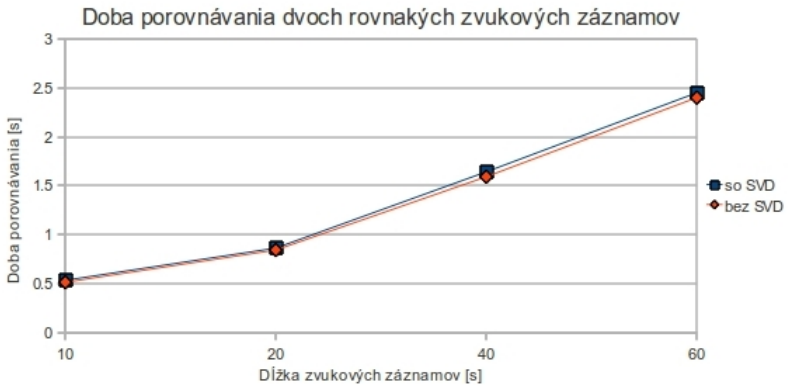
if $DIST(P1_{[3]}, P2_{[3]}) > H_1$ **then**
  return RÔZNE
end if
if $DIST(P1_{[12]}, P2_{[12]}) < H_2$ **then**
  return ROVNAKÉ
else
  return RÔZNE
end if

Dôvodom takéhoto spôsobu porovnávania je typ vybraného algoritmu diffu, ktorého časová zložitosť silne závisí od veľkosti zmeny v porovnávaných súboroch (viď nižšie). Preto je výhodnejšie použiť SVD k aproximácii získaných dát a pre testovanie rozdielu dvoch objektov použiť ich aproximáciu. Výhodnosť tohoto prístupu ilustrujú obrázky 6.2 6.3. Na obrázku 6.2 je zobrazená doba porovnania dvoch rôznych záznamov zvuku. Je vidieť, že použitím aproximácie algoritmom SVD sa značne vylepšil výkon aplikácie pre najhorší možný prí-

pad. Obrázok 6.3 ukazuje dobu porovnania dvoch rovnakých záznamov zvuku. Je vidieť, že výpočet SVD veľmi nespomalil aplikáciu v jej najlepšom možnom prípade.



Obrázek 6.2: Graf zobrazujúci dobu porovnávania dvoch rôznych zvukových záznamov aplikáciou audio diff bez použitia SVD a s použitím SVD.



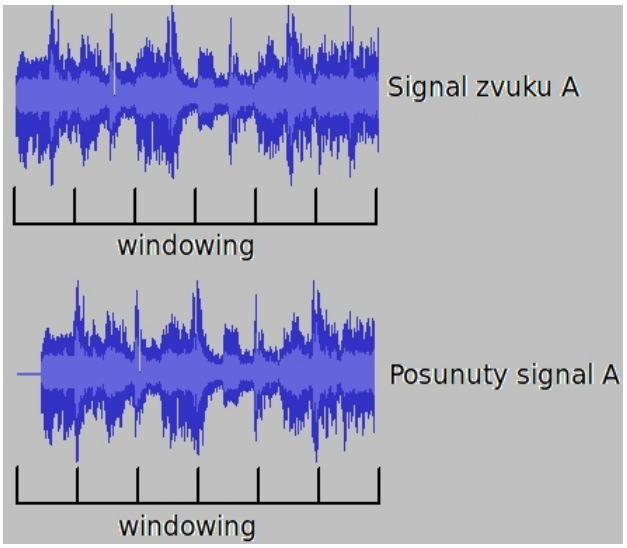
Obrázek 6.3: Graf zobrazujúci dobu porovnávania dvoch rovnakých zvukových záznamov aplikáciou audio diff bez použitia SVD a s použitím SVD.

Posledným komponentom je komponent vykonávajúci funkciu diff. Je pravda, že v jazyku python existuje už implementácia tohto algoritmu, avšak podmienkou jeho použitia je, aby nad dátami bolo možné použiť funkciu haš. Toto však nie je možné splniť, pretože by sa týmto spôsobom mohli znehodnotiť dáta získané extrakciou zo zvukového signálu, keďže tieto dáta sú viac dimenzionálne a ich zobrazenie do jednorozmerného priestoru použitím obyčajnej hašovacej funkcie by ich odstránilo vzájomné rozdiely. Z toho dôvodu bol implementovaný iný algoritmus, ktorý týmto neduhom netrpí. Algoritmus diffu bol prevzatý a upravený z existujúceho nástroja diff (GNU Diffutils), ktorý je založený na algoritme uvedenom v kapitole 4.4 a vydávaný pod GNU verejnou licenciou. V tomto algoritme boli však vykonané mnohé zmeny.

Pôvodný algoritmus pracuje s textovými dokumentami, a preto ako základnú jednotku porovnávania používa hodnoty hashu jednotlivých riadkov textových dokumentov. Bolo rozhodnuté, že implementovaný algoritmus musí byť nezmenený pri akejkoľvek obmene v spôsobe porovnávania dvoch základných jednotiek, prípadne aj zmeny typu porovnávaných jednotiek. Z tohto dôvodu bola vytvorená abstraktná trieda `Item` reprezentujúcu

základnú jednotku, s ktorou pracuje algoritmus. Okrem toho, že táto trieda poskytuje niektoré funkcie potrebné pre správne fungovanie algoritmu diffu, navyše obsahuje čisto virtuálny preťažený operátor porovnania ==, ktorý musí byť implementovaný potomkom tejto triedy. To znamená, že táto trieda slúži ako rozhranie pre objekty, ktoré dokáže implementovaný algoritmus spracovať. Takýmto spôsobom je zabezpečená jednoduchá zmena objektov, s ktorými výsledný algoritmus pracuje.

Ďalšou výraznou zmenou je spôsob porovnávania jednotlivých objektov. Tým, že je celý signál rozdelený na atomické jednotky porovnávania podľa toho, ako je rozdelený v procese windowing, vznikajú problémy so vzájomným posunutím porovnávaných signálov. Tento problém je ilustrovaný na obrázku 6.4. Signál A je zvuk hudby, posunutý signál A vznikol pridaním signálu s nulovou energiou (ticho) o dĺžke rovnej polovici veľkosti okna použitého v procese windowing pred začiatok samotného signálu. Je vidieť, že kvôli takémuto posunu sú posunuté aj počítané dáta, čo vedia k nesprávnym výsledkom.



Obrázek 6.4: Ilustrácia možného posunu dvoch rovnakých signálov.

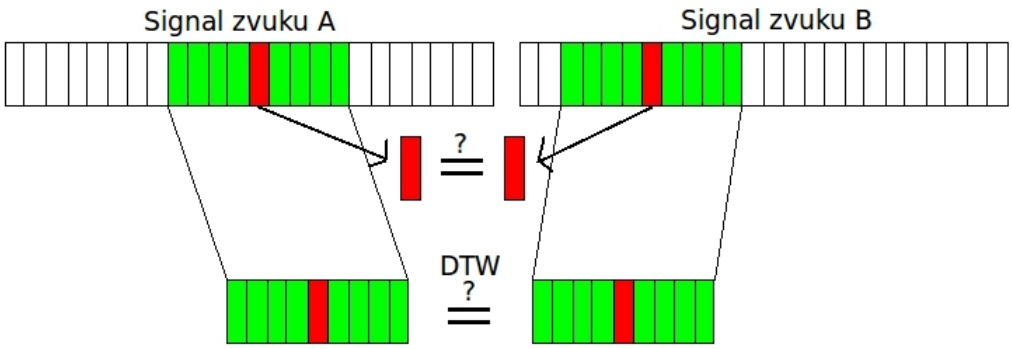
Podobný problém taktiež vzniká pri porovnávaní signálu zakódovaného bezstratovým kódovaním a jeho varianty kódovanej napríklad pomocou MPEG-1. V tomto prípade sú pri kódovaní MPEG-1 zahodené niektoré dáta, čo pri spätnej rekonštrukcii do formy zvukovej vlny vedie k nezrovnalostiam medzi pôvodným signálom a signálom kódovaným stratovou kompresiou. Toto taktiež vedie k nesprávnym výsledkom.

Na druhej strane, ak sú porovnávané dva rôzne zvukové signály, môže sa stať, že niektoré úseky ohraničené oknom budú veľmi podobné, pretože dĺžka týchto úsekov je malá. Napríklad je možné, aby dva úseky zvukového signálu o dĺžke 100ms boli takmer totožné, pokiaľ sa jedná napríklad o veľmi podobnú hudbu, prípadne podobné slová. Následkom by bolo síce správne porovnanie dvoch zvukov, avšak podobnosti dvoch audio signálov napríklad o dĺžke 100ms sú v praxi nepodstatné, pretože úlohou je vyhodnotiť podobnosti dvoch zvukových signálov vo väčšom meradle. Ľudským uchom nie je napríklad možné určiť, či 100ms zvuku je rovnakých alebo nie, pokiaľ neuvažujeme o extrémoch.

Aby bolo možné vyššie spomenuté problémy odstrániť, je nutné do porovnávania podobnosti jednotlivých vzoriek zahrnúť aj informáciu o podobnosti ich okolí. Preto bolo pri porovnávaní vzoriek použité DTW. Pri porovnávaní dvoch vzoriek A a B sa postupuje následne:

1. Index vzorky A je i, index B je j, index je poradie vzorky vo vektore reprezentujúcom signál,
2. vyhodnotí sa podobnosť vzoriek A_i a B_j,
3. pomocou DTW sa vypočíta vzdialenosť signálov v intervale (A_{i-h},A_{i+h}) a (B_{j-h},B_{j+h}), h je experimentálne určená skalárna hodnota definujúca hranicu okolia porovnávanej vzorky použitú v algoritme DTW,
4. pokiaľ A_i a B_j boli vyhodnotené ako rovnaké a vzdialenosť úsekov signálov vypočítaná pomocou DTW je menšia alebo rovná polovici dĺžky týchto úsekov, sú A_i a B_j vyhodnotené ako rovnaké, v opačnom prípade ako rôzne,
5. ak A_i a B_j boli vyhodnotené ako rôzne a vzdialenosť úsekov signálov vypočítaná pomocou DTW je väčšia alebo rovná polovici dĺžky týchto úsekov, sú A_i a B_j vyhodnotené ako rôzne, v opačnom prípade ako rovnaké.

Tento postup je taktiež ilustrovaný na obrázku 6.5, kde sú červenou farbou vyznačené aktuálne porovnávané vzorky a zelenou ich príslušné okolia použité v algoritme DTW.

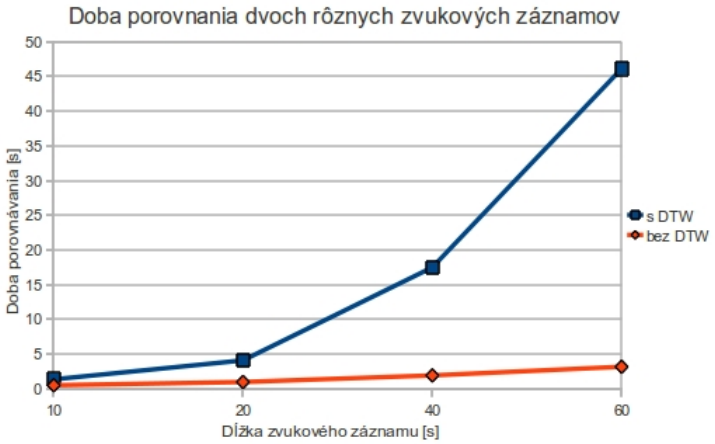


Obrázek 6.5: Postup pri porovnávaní dvoch vzoriek signálov A, B. Červenou sú vyznačené porovnávané vzorky, zelenou ich okolie pre DTW.

Porovnávanie dvoch vzoriek audio signálu takýmto spôsobom výrazne pomôže zlepšiť výstup audio diffu. Druhou stranou mince je však prílišná výpočtová náročnosť tohto spôsobu porovnávania. Toto je ilustrované na obrázku 6.6, kde je zobrazený čas potrebný k vyhodnoteniu dvoch rôznych audio súborov s použitím DTW a bez neho. Ako je vidieť doba vyhodnotenia rastie takmer s druhou mocninou dĺžky porovnávaných záznamov.

Avšak výstup bez použitia DTW nie z pohľadu požiadaviek na cieľovú aplikáciu vo väčšine prípadov správny, čo ilustrujú nasledujúce príklady porovnania audio záznamov aplikáciou audio diff:

- Porovnanie dvoch rovnakých zvukových záznamov o dĺžke 10s, kde jeden je posunutý o 50ms pridaním ticha na začiatok záznamu. V tomto prípade prístup k porovnávaniu pomocou DTW bol prospešný a výstup aplikácie bol prázdny, čo znamená, že oba porovnávané záznamy sú rovnaké. V prípade výstupov, ktoré boli z priestorových dôvodov skrátené, sú uvedené len prvé štyri riadky a toto skrátenie je naznačené znakom ....



Obrázek 6.6: Graf zobrazujúci dobu porovnávania dvoch rôznych zvukových záznamov aplikáciou audio diff bez použitia DTW a s použitím DTW.

```
DTW použité:  Bez použitia DTW:
              0.10,0.30c0.10,0.30
              0.40,0.70c0.40,0.70
              0.80,0.90c0.80,0.90
              1.00,1.20c1.00,1.20
              ...
```

- porovnanie dvoch rovnakých zvukových záznamov o dĺžke 60s, kde jeden je posunutý o 50ms pridaním ticha na začiatok záznamu. Je vidieť, že v prípade použitia DTW bolo zle vyhodnotených len 200ms, výstup bez použitia DTW je nesprávny,

```
DTW použité:              Bez použitia DTW:
14.30,14.50c14.30,14.50   0.10,0.30d0.10
                          0.40,0.60c0.30,0.50
                          1.50,1.70c1.40,1.60
                          2.70,2.90c2.60,2.80
                          ...
```

- porovnanie dvoch rovnakých zvukových záznamov o dĺžke 10s, kde jeden je kódovaný pomocou MPEG-1. DTW znovu pomohlo značne zlepšiť výsledok a zle určených bolo spolu len 400ms,

```
DTW použité:          Bez použitia DTW:
2.80,3.00c2.80,3.00   0.50,0.70c0.50,0.70
6.00,6.20c6.00,6.20   1.00,1.10c1.00,1.10
                      1.20,1.40c1.20,1.40
                      1.50a1.50,1.70
                      ...
```

- porovnanie dvoch rôznych zvukových záznamov o dĺžke 10s. V tomto prípade je výsledok s použitím DTW ideálny.

```
DTW použité:            Bez použitia DTW:
0.00,10.00c0.00,10.00   0.00a0.00,0.90
                        0.10,0.90c0.90,1.40
                        1.00,1.40c1.50,2.40
                        1.50,3.20c2.50,2.80
                        ...
```

Kapitola 7

Výber audio knižnice a jej nastavení

Pokiaľ by nebolo možné rozlíšiť riadky v textovom súbore, akokoľvek dobrý algoritmus diffu by nepracoval správne. To isté platí aj v tomto prípade. Ak nie je možné rozlíšiť rozdiely medzi dvoma zvukovými signálmi, celá vytváraná aplikácia nebude fungovať. Z tohoto dôvodu je najdôležitejšou časťou aplikácie vhodná reprezentácia zvukového signálu. Za túto reprezentáciu je zodpovedná audio knižnica, ktorá extrahuje charakteristické znaky zo signálu a tiež nastavenia použité pri tejto extrakcii. Taktiež extrakcia týchto znakov bude podstatným spôsobom vplývať na výkon aplikácie. Preto bol kladený veľký dôraz na výber vhodnej knižnice a tiež nastavení pri extrakcii charakteristických vlastností zvukového signálu. Ako sa ukázalo, toto však bola neľahká úloha, pretože vybrať nastavenia, ktoré fungujú rovnako dobre pre reč, hudbu a iné zvuky je prinajmenšom náročné. Výberu knižnice a nastavení predchádzal celý rad pokusov, na ktorého konci bola vybraná najvhodnejšia audio knižnica a jej nastavenia pri extrakcii charakteristických čŕt signálu zvuku. TODO popis kapitol V tejto kapitole

7.1 Výber audio knižnice

K dispozícii boli dva nástroje slúžiace na extrakciu charakteristických znakov signálu zvuku. Prvý nástroj sa nazýva HTK a druhý Kaldi. Podrobnejšie popisy týchto nástrojov sú uvedené v kapitole 5.1. Oba knižnice majú podobné vlastnosti a tiež podobnú množinu funkcií, ktoré poskytujú. V skutočnosti, v implementácii tejto aplikácie bude používaný len zlomok funkcionality vybranej knižnice, a to len funkcie pre extrakciu čŕt zvuku a získania aproximácie dát pomocou SVD. Pri rozhodovaní, ktorý nástroj bude použitý, boli brané do úvahy nasledujúce kritériá:

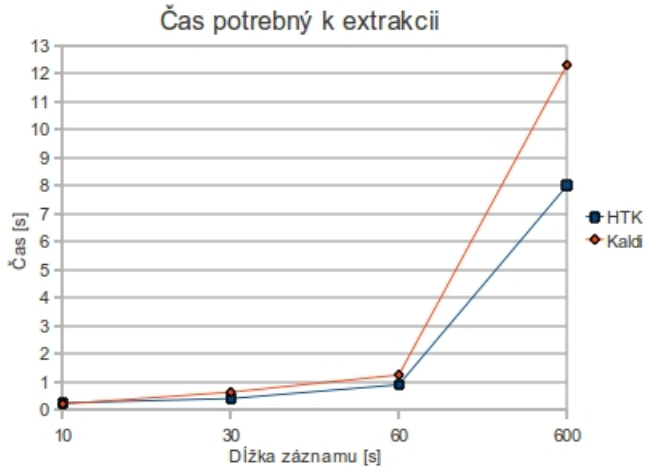
- rýchlosť extrakcie
- jednoduchosť použitia
- licenciu

Keďže medzi požiadavky na vytváraný nástroj audio diffu patrí rozumná rýchlosť získania výsledkov, má rýchlosť s akou nástroj dokáže počítať požadované koeficienty cepstra mel-frekvencie najvyššiu prioritu. Z tohto dôvodu som bol vykonaný experiment, v ktorom sú porovnávané časové požiadavky oboch nástrojov.

Boli vytvorené štyri nahrávky audia s rôznou dĺžkou. Každý nástroj desať krát počítal koeficienty z každej nahrávky, pričom bol meraný čas, ktorý bol nutný k výpočtu. Následne

bol vypočítaný priemerný čas, potrebný pre získanie koeficientov pre každú nahrávku pre oba nástroje. K meraniu času bol použitý linuxový nástroj `time`. Výsledky z meraní sú zobrazené na obrázku grafu 7.1. Na vertikálnej ose je uvedený priemerný čas v sekundách potrebný pre výpočet (súčet užívateľského a systémového času), na horizontálnej ose je uvedená dĺžka nahrávky v sekundách.

Nastavenia oboch nástrojov boli samozrejme rovnaké. Oba extrahovali dvanásť koeficientov a koeficient energie. Bolo použité hammingovo okno s veľkosťou 20ms a posunom 10ms. V oboch prípadoch bol použitý rovnaký počet 25 trojuholníkových filtrov. Nebola vykonaná preemfáza a bol odstránený DC ofset. Nahrávky mali vzorkovaciu frekvenciu 44100Hz. Z výsledkov je vidieť, že nástroj HTK potrebuje o niečo menej času k výpočtu, čo sa prejavilo najmä pri nahrávke s dĺžkou 600s.



Obrázek 7.1: Čas potrebný knižnicami HTK a Kaldi k extrakcii MFC koeficientov.

K ovládaniu nástroja HTK je nutné vytvoriť textový konfiguračný súbor, kde sú uvedené nastavenia rôzne od východzích. Tento konfiguračný súbor je pri spustení nástroja načítaný a sú nastavené príslušné parametre. To znamená, že zmeny v nastavení extrakcie charakteristických znakov zvuku je nutné zapísať do textového súboru, čo môže byť obmedzujúce. Taktiež by to bolo problematické v prípade, ak bude nutné zmeniť nastavenia podľa parametrov audia (napríklad jeho vzorkovacej frekvencie).

Zmeny východzích nastavení v druhom nástroji sa nastavujú ako parametre pri jeho spustení, navyše ako bolo uvedené v kapitole 6.2 bol pre prácu s touto knižnicou vytvorený adpatér, ktorý priamo používa jej funkcie . Čo sa týka ovládania jednotlivých nástrojov je podľa názoru autora flexibilnejší druhý nástroj.

Vo svojich možnostiach sú si oba nástroje podobné (aspoň čo sa týka, možností pre extrakciu koeficientov cepstra mel-frekvencie). Avšak knižnica Kaldi má možnosť použiť pri extrakcii o jeden typ okna viac ako HTK. V HTK je možné zvoliť či použiť alebo nepoužiť hammingovo okno. Informácie o tom aké okno je nastavené v prípade nepoužitia hammingovho neboli dostupné. Táto skutočnosť by bola problémová v prípade ďalších experimentov s nastaveniami knižnice. S Kaldi je možné použiť hammingovo, hannovo a obdĺžnikové okno.

Problém však nastáva v prípade licencie HTK, ktorá neumožňuje voľnú distribúciu zdrojových kódov tretím stranám. Keďže aplikácia audio diff je vytváraná ako open source projekt, nie je možné HTK použiť pre túto aplikáciu. To znamená, že v implementácii výslednej aplikácie audio diff a tiež v experimentoch s nastaveniami parametrov extrakcie MFC koe-

ficientov bude použitá knižnica Kaldi.

7.2 Výber parametrov extrakcie

V tejto časti budú ukázané a popísané experimenty na základe ktorých, boli vybrané vhodné nastavenia knižnice pre extrakciu charakteristických rysov signálu zvuku. Ako bolo uvedené v kapitole 6.1 na jeho reprezentáciu budú použité koeficienty cepstra mel-frekvencie. To znamená, že ak v nasledujúcom texte budú spomínané charakteristické znaky či rysy audio záznamu, bude sa jednať o hodnoty koeficientov cepstra mel-frekvencie.

Keďže podľa dostupných informácií nebola ešte použitá podobná aplikácia MFC koeficientov, bolo nutné nájsť vhodné nastavenia extrakcie rysov. Kvôli tomu, aby bolo vidieť akým spôsobom vplývajú zvyčajné zmeny v audio signály ako šum, či kódovanie stratovou kompresiou na hodnoty extrahovaných MFC koeficientov, bolo vykonaných niekoľko experimentov. Účelom týchto experimentov je výber vhodných nastavení a získanie čo najlepšej aproximácie ľudského sluchového zážitku z daného zvukového signálu. To znamená, že pokiaľ sú zmeny vo zvuku nepočuteľné hodnoty extrahovaných rysov by mali byť podobné. V opačnom prípade by mali byť čo najodlišnejšie. Príkladom môžu byť dva audio záznamy, ktoré obsahujú rovnaký zvuk, avšak jeden obsahuje viac šumu ako druhý. V takomto prípade musia byť hodnoty MFC koeficientov podobné.

Na účely experimentoch bolo vytvorených niekoľko špecifických audio záznamov, ktoré sú použité pre simuláciu vyššie uvedených poškodení a úprav. Popis audio záznamov a spôsob ich získania je uvedený nižšie v časti 7.2.1.

Problémom je, že dva signály zvuku o dĺžke jedného okna (napríklad 20ms), nie je možné uchom spoľahlivo rozlíšiť a nie je možné povedať či sú tieto signály rovnaké alebo nie. To znamená, že pre porovnanie dvoch zvukov je nutné vypočuť dlhšie úseky. V tomto prípade je však signál zvuku reprezentovaný vektorom vektorov koeficientov, kde jeden vektor koeficientov reprezentuje určitú časť signálu. Dĺžka tejto časti je závislá na veľkosti použitého okna. Keďže signál zvuku hudby či reči nie je konštantný a dochádza k jeho zmene medzi jednotlivými časťami ohraničenými pomocou okna v procese windowing, je nutné zvoliť spôsob akým spôsobom bude určovaná podobnosť dvoch audio záznamov.

Vzdialenosť medzi dvoma vektormi MFC koeficientov je počítaná rovnako ako v implementácii (viď 6.1). Najmenší či najväčší rozdiel vektorov MFC koeficientov dvoch záznamov zvuku, by bol príliš náchylný k extrémom. Z tohto dôvodu je v experimentoch sledovaná priemerná hodnota vzdialenosti vektorov MFC koeficientov originálneho záznamu zvuku a jeho poškodeného či upraveného náprotivku. Matematická definícia priemernej vzdialenosti:

$$DIST_{AVG}((i_1, i_2, \ldots, i_n), (j_1, j_2, \ldots, j_n)) = \frac{\sum_{k=1}^{n} DIST(i_k, j_k)}{n} \tag{7.1}$$

kde i,j sú vektory MFC koeficientov a n je počet týchto vektorov a v nasledujúcom texte bude táto priemerná vzdialenosť označovaná už len ako vzdialenosť dvoch zvukových záznamov.

Pri výbere nastavení sa začalo s nastavením tradičným pre oblasť rozpoznávania reči a úpravou týchto nastavení na ad hoc hodnoty boli hľadané nastavenia, ktoré dávajú najlepšie výsledky zodpovedajúce tomuto cieľu. Samozrejme je snahou získať tieto výsledky s čo najmenšími výpočtovými a časovými nárokmi.

7.2.1 Popis testovacích záznamov

Bolo vytvorených 5 typov vzoriek audio záznamov. Na týchto záznamoch sa nachádza hudba, reč a časti s úplným tichom. Výsledné záznamy vznikli kombináciou týchto elementov pomocou nástroja Audacity. Hudba bola získaná z voľne dostupných zdrojov z internetu so vzorkovacou frekvenciou 44100Hz. Následne boli tieto hudobné záznamy prevedené zo stereo módu na mono za použitia Audacity. Túto hudbu je možné zaradiť do rockového žánru. Hlasové záznamy boli opäť získané z voľne dostupných zdrojov z internetu avšak so vzorkovacou frekvenciou len 22050Hz.

Časti ticha boli vytvorené pomocou nástroja Audacity. Všetky audio záznamy boli vytvorené vo vzorkovacích frekvenciách 8kHz, 16kHz a 44100Hz. Tieto frekvencie neboli použité náhodou, ale každá má svoj význam. Frekvencia 8kHz je používaná v telefónnych zariadeniach, 16kHz je často používaná v oblasti rozpoznávania reči a frekvencia 44100Hz je štandardná pre audio záznamy hudby uloženej na kompaktných diskoch (CD). Všetky záznamy majú bitovú hĺbku o veľkosti 16b. Originálne audio záznamy boli vytvorené pomocou Audacity vo vzorkovacej frekvencii 44100Hz a následne prevzorkované do ostatných frekvencií. To znamená, že bolo nutné reč, ktorá bola získaná pôvodne vo vzorkovacej frekvencii 22050Hz, prevzorkovať (taktiež s Audacity) na frekvenciu 44100Hz. Nasledujúci zoznam popisuje vytvorené vzorky záznamov zvuku použitých pri testovaní (v hranatej zátvorke je uvedený názov vzorky, pod ktorou vystupuje vo výsledkových tabuľkách):

- Prvý záznam pozostáva z desiatich sekúnd rockovej hudby [zaznam1],
- druhý pozostáva z dvadsiatich sekúnd reči v anglickom jazyku [zaznam2],
- tretí obsahuje šesťdesiat sekúnd rockovej hudby. Táto nahrávka bola vytvorená, aby bolo možné sledovať správanie extrahovaných MFC koeficientov na dlhších záznamoch. Záznamy tejto dĺžky sa ukážu podstatné najmä v kapitole 7.2.4 [zaznam3],
- štvrtý záznam pozostáva zo striedania ticha a rockovej hudby. Vznikol tak, že sa pridali úseky ticha do tretieho záznamu. Táto nahrávka bola vytvorená, aby bolo možné sledovať extrahované znaky zvuku v prítomnosti častí s absolútnym tichom [zaznam4],
- posledný, piaty záznam, pozostáva z reči, ticha a rockovej hudby a mal by aproximovať napríklad zvuk filmu, kde sa často tieto typy zvuku striedajú. Vznikol kombináciou predošlých záznamov [zaznam5].

Diametrálny rozdiel medzi tvarom signálov hudby a reči ilustruje obrázok 7.2, kde sú zobrazené príklady zvukových vĺn originálnych záznamov.

Následne boli tieto audio záznamy prevzorkované na spomínané nižšie frekvencie. Takto vytvorené audio súbory budú v texte označované ako originálne záznamy. Je samozrejmé, že po prevzorkovaní na nižšie frekvencie sa zmenil zvuk najmä častí s hudbou, čo však nie je problém, pretože v experimentoch sa budú vzájomne porovnávané len audio záznamy s identickou vzorkovacou frekvenciou.

Ďalej bolo vytvorených niekoľko typov poškodení a úprav originálnych audio záznamov. Nové upravené záznamy vznikli troma spôsobmi:

- pridaním šumu
- znížením hlasitosti

Obrázek 7.2: Zvukových vlny hudby a reči.

- zakódovaním do formátu MPEG-1

Prvá skupina upravených audio záznamov vznikla pridaním bieleho šumu do originálnych záznamov v nástroji Audacity. Táto skupina pozostávala z troch typov podskupín v závislosti na množstve pridaného šumu. Nasledujúca tabuľka 7.1 zobrazuje popis a subjektívne hodnotenie týchto nahrávok.

Množstvo pridaného šumu	*Subjektívne hodnotenie výslednej nahrávky*
globálny pomer signálu k šumu (SNR) výslednej nahrávky = 30DB	Pridaný šum nie je počuteľný v nahrávkach s hudbou. V častiach s tichom a rečou je síce počuť slabý šum, avšak celkovo môžu byť takto upravené záznamy vyhodnotené ako totožné s originálom.
globálne SNR výslednej nahrávky = 50DB	Žiadny rozdiel od originálnych nahrávok nie je možné počuť v častiach s hudbou a rečou. V častiach s tichom je počuť len minimálny šum, ktorý môže byť považovaný za ticho.
globálne SNR výslednej nahrávky = 70DB	V tomto prípade sú výsledné nahrávky takmer totožné s originálnymi záznamami.

Tabulka 7.1: Popis nahrávok s pridaným bielym šumom.

Druhá skupina upravených záznamov bola vytvorená pridaním ružového šumu (má menej energie vo vyšších frekvenciách ako biely šum) do originálnych záznamov. Tabuľka 7.2 zobrazuje popis a subjektívne hodnotenie týchto nahrávok.

Množstvo pridaného šumu	*Subjektívne hodnotenie výslednej nahrávky*
globálne SNR výslednej nahrávky = 30DB	V týchto nahrávkach je počuť mierny šum, avšak nie dostatočne veľký, aby mohli byť považované za odlišné od originálov.
globálne SNR výslednej nahrávky = 10DB	Nahrávky sú silne zašumené a nie je možné ich považovať za podobné originálnym záznamom.

Tabulka 7.2: Popis nahrávok s pridaným ružovým šumom.

Ďalšia skupina vznikla prekódovaním originálnych záznamov do formátu MPEG-1 pri zachovaní konštantnej vzorkovacej frekvencie a bitového pomeru 128kbps a následným prekódovaním do formátu wav, keďže vybraná knižnica Kaldi vie pracovať len s nekomprimovaným formátom wav a v praxi bude nutné všetky formáty audio nahrávok prekódovať do tohto formátu.

Týmto postupom však došlo k predĺženiu takto vytvorených záznamov o niekoľko vzoriek, čo znamená extrakciu väčšieho množstva vektorov MFC koeficientov ako z originálnych záznamov. Počítanie vzdialenosti v testoch bude v tomto prípade vykonané len na počte vektorov extrahovaných z originálneho audio záznamu a nadbytočné rámce budú jednoducho ignorované.

Pri tejto skupine záznamov nie sú počuteľné rozdiely v žiadnych častiach zvuku pri vzorkovacích frekvenciách 16kHz a 44100Hz, avšak pri vzorkovacej frekvencii 8kHz v častiach s rečou a hudbou dochádza pri zakódovaní a následnom dekódovaní k poškodeniu audio záznamu a je možné počuť podobné artefakty ako pri prehrávaní hudby z poškodeného kompaktného disku. Z tohoto dôvodu nie je možné pri porovnávaní považovať tieto záznamy za nezmenené a je žiaduce, aby bola ich vzdialenosť od originálov čo najväčšia.

Posledná skupina upravených audio záznamov vznikla znížením hlasitosti originálnych audio záznamov o 25dB. Keďže zmena hlasitosti nie je zmenou zvuku, ktorú je možné počuť, respektíve je možné ju upraviť jednoduchým zvýšením hlasitosti, sú tieto záznamy považované za nezmenené.

V experimentoch sa vždy počíta vzdialenosť medzi originálnymi záznamami a ich príslušnými upravenými či poškodenými verziami v rovnakej vzorkovacej frekvencii.

Okrem vyššie spomenutých originálnych záznamov a ich poškodených variantov bolo vytvorených ďalších päť záznamov, ktoré budú v testoch použité ako referenčné záznamy. Vzdialenosť týchto záznamov od originálov musí byť vždy väčšia ako vzdialenosť poškodených záznamov, kde toto poškodenie nie je počuteľné. Toto je základná podmienka pre všetky nastavenia systému. Keďže štvrtý a piaty záznam boli vytvorené len preto, aby bolo možné sledovať správanie extrahovaných charakteristických znakov audio záznamu pozostávajúceho z viacerých typov zvuku, nie je k nim nutné vytvárať vyššie spomenuté referenčné záznamy.

To znamená, že boli vytvorené spolu štyri referenčné záznamy obsahujúce hudbu s dĺžkou trvania desať a šesťdesiat sekúnd, a jeden záznam reči s dĺžkou trvania dvadsať sekúnd. Opäť sú tieto záznamy vytvorené v troch vzorkovacích frekvenciách podobným spôsobom ako prvých päť originálnych záznamov. Obsahujú hudbu získanú z voľne dostupných zdrojov na internete so vzorkovacou frekvenciou 44100Hz a reč so vzorkovacou frekvenciou 22050Hz, ktorá bola prevzorkovaná na 44100Hz. Záznamy obsahujú nasledujúci zvuk (v hranatej zátvorke je uvedený názov vzorky, pod ktorou vystupuje vo výsledkových tabuľkách):

- 10 sekúnd klasickej hudby [zaznam6],
- 10 sekúnd rockovej hudby (iná ako v prípade originálnych záznamov) [zaznam7],
- 60 sekúnd klasickej hudby [zaznam8],
- 60 sekúnd rockovej hudby [zaznam9],
- 20 sekúnd sekúnd reči v anglickom jazyku (odlišná od reči použitej v originálnom zázname s rečou) [zaznam10].

V experimentoch budú počítané vzdialenosti nahrávok obsahujúce hudbu od originálneho záznamu obsahujúceho hudbu rovnakej dĺžky, rovnako bude počítaná vzdialenosť referenčného záznamu reči od originálnej nahrávky s rečou.

Aby som sa vyhol skresleniu výsledkov kvôli zlému výberu referenčných audio záznamov najmä v oblasti porovnania hudby (Napríklad zlá voľba žánru prípadne audio skladieb) vytvoril som ďalšie štyri testovacie audio záznamy. Dva z týchto záznamov obsahujú šesťdesiat sekúnd hudby spadajúcej do žánru klasickej hudby. Ďalšie dva obsahujú šesťdesiat sekúnd agresívnej rockovej hudby. Opäť bolo toto audio získané z internetu so vzorkovacou frekvenciou 44100Hz a následne prevzorkované do 16kHz a 8kHz. Vo výsledkových tabuľkách bude vypočítaná vzdialenosť dvoch záznamov s rockovou hudbou označená ako "rock", podobne vzdialenosť záznamov s klasickou hudbou ako "klasika". Cieľom nasledujúcich experimentov je nájsť také nastavenia extrakcie, aby vzdialenosť záznamov s pridaným šumom s globálnym SNR o hodnote 30dB, 50dB a 70dB, záznamov so zníženou hlasitosťou a záznamov zakódovaných do formátu MPEG-1 od originálov bola čo najmenšia a vzdialenosť záznamov s pridaným šumom s globálnym SNR o hodnote 10dB, referenčných záznamov a testovacích záznamov bola čo najväčšia. Z úsporných dôvodov nebolo možné do tejto práce vložiť všetky získané údaje, preto sa v týchto experimentoch nachádzajú len niektoré hodnoty vzdialeností, ktoré by mali ukázať tendenciu extrahovaných dát.

7.2.2 Nastavenie typu okna

Keďže existuje mnoho typov okien používaných pri spracovaní signálov, prvým krokom pri extrakcii rysov audio záznamu je výber vhodného okna. Na základe tohoto experimentu bude vybratý typ okna. Okrem vzdialenosti záznamov bude v tabuľkách uvedený v percentách aj počet vektorov MFC koeficientov so vzdialenosťou menšou ako celková priemerná vzdialenosť nahrávok. Tento údaj je zisťovaný kvôli vytvoreniu celkového obrazu o rozložení hodnôt vzdialeností jednotlivých vektorov, ako napríklad či bola výsledná vzdialenosť ovplyvnená nejakými extrémnymi hodnotami.

V experimente boli počítané vzdialenosti pomocou troch typov okien, a to obdĺžnikového okna, hammingovho okna a hannovho okna. Spoločné nastavenia pre extrakciu boli nasledovné:

- Pre každé okno bolo extrahovaných 12 MFC koeficientov plus koeficient energie,
- nebola vykonaná preemfáza,
- z nahrávok bol odstránený DC ofset,
- cepstrálny zdvih bol nastavený na hodnotu 22,
- frekvencia zvuku bola zdola obmedzená na 20Hz a zhora na 16kHz pre záznamy frekvenciou 44100Hz. V ostatných prípadoch len zdola na 20Hz,
- počet trojuholníkových filtrov bol nastavený na 25,
- nebol použitý dithering a hodnoty koeficientov z okien obsahujúcich nulovú energiu (ticho) boli nastavené na nulu. Toto bolo vykonané z dôvodu čo najmenšej variácie výsledných hodnôt a pridanie jednotkového šumu (dithering) bolo považované za príliš veľký zásah do zvuku audio záznamu.

Extrakcia bola prevedená pre tri typy okien:

- obdĺžnikové, dĺžka okna bola 20ms s posunom rovným veľkosti jedného okna, čiže 20ms. Pretože kvôli tvaru okna by boli hodnoty koeficientov skreslené v prípade, ak by sa dve nasledujúce okná prekrývali, boli radšej akceptované nezrovnalosti signálu na krajoch okien,
- hammingovo, dĺžka okna bola 20ms s posunom 10ms,
- hannovo, dĺžka okna bola 20ms s posunom 10ms.

Všetky dáta získané pri tomto experimente sa nachádzajú v prílohe B. Pretože existovali pochybnosti, či energia signálu zvuku bude pre jeho diskrimináciu potrebná, bola počítaná vzdialenosť nahrávok s energiou aj bez energie. Po získaní výsledkov sa táto pochybnosť potvrdila. Veľkosti vzdialenosti s energiou a bez nej boli veľmi podobné a v prípade záznamov so zníženou hlasitosťou boli vzdialenosti počítané s energiou prirodzene väčšie ako bez nej. Najviac sa vzdialenosti s energiou a bez nej líšili pre originálne nahrávky obsahujúce veľa ticha a ich verzie s pridaným šumom (pre ilustráciu tabuľka 7.3). Z tohoto dôvodu bolo rozhodnuté, že pre diskrimináciu signálu energia použitá nebude.

	zaznam5			
	s energiou		**bez energie**	
typ okna	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
obdĺžnikové	169.56	75.62	229.19	70.54
hammingovo	458.43	77.60	518.01	68.12
hannovo	1513.46	80.35	1573.04	80.32

Tabulka 7.3: Porovnanie vzdialenosti počítanej s a bez energie. Uvedená nahrávka je verzia záznamu v texte označenom ako zaznam5 s globálnym SNR rovným 30dB (pridaný ružový šum, vzorkovacia frekvencia 44100Hz).

Na základe výsledkov je tiež možné povedať, že dĺžka upraveného záznamu nemá vplyv na jeho vzdialenosť od originálu a tým pádom ani na extrahované MFC koeficienty. Taktiež vzdialenosť upravených záznamov poskladaných z viacerých typov zvuku približne zodpovedá vzdialenostiam nahrávok obsahujúcich jednotlivé zložky zvuku (pre ilustráciu viď tabuľku 7.4).

	zaznam1		**zaznam3**	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
obdĺžnikové	0.398	94.2	0.297	94.967
hammingovo	1.07	90.991	0.944	93.949
hannovo	1.251	91.291	1.113	94.016

Tabulka 7.4: Porovnanie vzdialenosti nahrávok s rôznou dĺžkou s rovnakým poškodením (pridaný biely šum, globálne SNR 50dB, vzorkovacia frekvencia 44100Hz)

Celkovo extrahované MFC koeficienty korelovali so sluchovým zážitkom. To znamená, že vzdialenosti upravených nahrávok od originálov, kde zmeny počuteľné neboli, boli oproti referenčným záznamom malé. Toto platí pre záznamy s pridaným šumom, kde viac pridaného šumu znamenalo väčšie vzdialenosti. Tabuľka 7.5 zobrazuje vzdialenosti dvoch vybraných záznamov s pridaným šumom s hodnotou globálneho SNR 30dB. Stĺpec počet určuje percento vektorov MFC koeficientov so vzdialenosťou nižšou ako priemerná.

	zaznam1		zaznam2	
typ okna	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
obdĺžnikové	28.49	83.20	734.36	69.10
hammingovo	60.00	81.98	2275.78	57.02
hannovo	63.91	82.28	7963.74	37.66

Tabulka 7.5: Vzdialenosti vybraných záznamov s globálnym SNR rovným 30dB (pridaný biely šum, vzorkovacia frekvencia 44100Hz).

Ostatné záznamy s menším množstvom pridaného šumu mali samozrejme menšie hodnoty vzdialeností. Taktiež jeho typ mal vplyv na tieto hodnoty, pričom pri nahrávkach s pridaným ružovým šumom boli nižšie. Taktiež podstatne nižšie vzdialenosti mali záznamy s pridaným šumom v nižších vzorkovacích frekvenciách. Príkladom môže byť tabuľka 7.6, kde si je možné všimnúť podstatné rozdiely vo vzdialenosti oproti tabuľke 7.5 najmä v prípade záznamu reči. Toto autor pripisuje faktu, že biely a ružový šum obsahujú pomerne veľa energie vo vyšších frekvenciách, ktoré nie sú pri nižšej vzorkovacej frekvencii zaznamenané.

	zaznam1		zaznam2	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
obdĺžnikové	29.00	84.17	365.97	75.48
hammingovo	72.66	80.06	671.18	66.82
hannovo	78.92	80.26	807.44	63.66

Tabulka 7.6: Vzdialenosti vybraných záznamov s globálnym SNR rovným 30dB (pridaný biely šum, vzorkovacia frekvencia 8kHz).

Záznamy so zníženou hlasitosťou mali vzdialenosti dokonca menšie ako zašumené záznamy so SNR rovným 50dB. Záznamy kódované do formátu MPEG-1 (tabuľka 7.7) mali podobnú vzdialenosť ako ako záznam s rečou s globálnym SNR o hodnote 30dB. Čo sú však pomerne vysoké hodnoty na to, že tieto záznamy zneli rovnako ako originály.

	zaznam1		zaznam2	
typ okna	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
obdĺžnikové	650.80	67.20	1177.87	64.69
hammingovo	995.10	64.76	1876.42	64.29
hannovo	1035.55	65.26	2054.68	64.99

Tabulka 7.7: Vzdialenosti vybraných záznamov kódovaných do formátu MPEG-1 (vzorkovacia frekvencia 44100Hz).

Problém nastáva v prípade nižších vzorkovacích frekvencií, kedy hoci rozdiel medzi záznamami zakódovanými do formátu MPEG-1 počuť taktiež nie je, sú priemerné vzdialenosti týchto záznamov od originálov príliš veľké. Túto situáciu ilustruje tabuľka 7.8, kde sú porovnané vzdialenosti záznamu obsahujúceho reč kódovaného do MPEG-1 v dvoch rôznych vzorkovacích frekvenciách. Je vidieť, že záznamy vo frekvencii 44100Hz majú vzdialenosť skoro o polovicu menšiu.

Pre porovnanie sú v tabuľke 7.9 uvedené vzdialenosti vybraných referenčných záznamov a v tabuľke 7.10 vzdialenosti testovacích nahrávok. Z týchto údajov je vidieť, že MFC koeficienty strácajú svoju diskriminačnú schopnosť signálu pri agresívnejšom type hudby, kde

	zaznam2			
	frekvencia 44100Hz		frekvencia 16kHz	
typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	1177.878	64.694	2130.568	57.573
hammingovo	1876.424	64.293	3471.494	56.849
hannovo	2054.686	64.995	3721.545	56.648

Tabulka 7.8: Porovnanie vzdialeností záznamu s rečou kódovaného do formátu MPEG-1 v rôznych frekvenciách.

vzdialenosti testovacích záznamov v stĺpci "rock" majú menšiu vzdialenosť ako vzdialenosť nahrávky reči s globálnym SNR 30dB.

	zaznam6		zaznam7	
typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	2118.39	58.80	2444.10	55.60
hammingovo	3371.74	59.96	3236.03	58.35
hannovo	3536.22	58.95	3288.10	58.05

Tabulka 7.9: Vzdialenosti vybraných referenčných záznamov (vzorkovacia frekvencia 44100Hz).

	klasika		rock	
typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	1545.22	55.63	1422.70	57.46
hammingovo	2180.17	55.62	1761.26	57.51
hannovo	2279.73	55.80	1795.69	57.56

Tabulka 7.10: Vzdialenosti kontrolných záznamov (vzorkovacia frekvencia 44100Hz).

Ďalej je vidieť, že podstatný vplyv na vzdialenosti nahrávok má aj typ zvuku. Šum sa na extrahovaných MFC koeficientoch prejavil najmä pri nahrávkach s rečou. V prípade záznamov s hubou boli vzdialenosti podstatne menšie ako pre reč. To môže byť problém pri vyhodnocovaní podobnosti signálov zvuku, keďže pri rovnakom poškodení sa získavajú rôzne hodnoty v závislosti na type zvuku. Príkladom môže byť nahrávka hudby s globálnym SNR 30db, kde je jeho vzdialenosť podstatne menšia ako v prípade rovnako poškodenej nahrávky reči. Táto skutočnosť je problémová, pretože nahrávka s globálnym SNR 10dB (tabuľka 7.11) má príliš malú hodnotu vzdialenosti, hoci jej zvuk je odlišný od originálu a jej vzdialenosť by mala byť čo najväčšia. Dokonca je pre všetky použité typy okien približne dva krát menšia ako v prípade záznamov hudby kódovaných do formátu MPEG-1, pričom tieto záznamy majú zvuk totožný s originálnymi verziami. Tento problém bude teda nutné riešiť inými nastaveniami.

Ďalšie zistenie bolo, že počet vektorov MFC koeficientov extrahovaných pomocou hannovho bol vo väčšine prípadov nižší ako päťdesiat percent. Toto naznačuje značné odchýlky extrahovaných hodnôt. Taktiež sú takto vypočítané koeficienty veľmi náchylné na zmenu hlasitosti záznamu. Napríklad vzdialenosť záznamu reči so zníženou hlasitosťou je dokonca vyššia ako vzdialenosť jedného z referenčných záznamov. Z tohto dôvodu nie je použitie hannovho okna vhodné.

	zaznam1		zaznam2	
typ okna	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
obdĺžnikové	269.28	69	1246.09	59.30
hammingovo	469.05	67.16	3644.13	59.43
pridaný hannovo	486.37	67.06	9975.32	41.52

Tabulka 7.11: Vzdialenosti vybraných záznamov s globálnym SNR rovným 10dB (pridaný ružový šum, vzorkovacia frekvencia 44100Hz).

Keďže rozdiel vzdialeností upravených záznamov s nezmeneným zvukom a referenčných nahrávok bol pomerne malý, bol vykonaný ďalší experiment, v ktorom sa k dvanástim koeficientom a energii pridali ich delty prvého rádu. Takto by mala byť dodaná určitá dynamická informácia.

Po získaní výsledkov tohoto experimentu bolo možné konštatovať, že delty prvého rádu nepomohli podstatne zväčšiť tento rozdiel. Skôr práve naopak. Získané dáta sa však kvôli priestorovým dôvodom v tejto práci neuvádzajú.

Na základe výsledkov týchto experimentov bolo rozhodnuté, že pre extrakciu bude použité obdĺžnikové okno a z jedného okna sa bude extrahovať len 12 MFC koeficientov bez delt a bez energie.

7.2.3 Nastavenie dĺžky okna

Cieľom tohoto experimentu je určiť najvhodnejšiu dĺžku okna pre extrakciu charakteristických znakov audio záznamu. Nastavenia knižnice sú rovnaké ako v predchádzajúcom experimente s tým, že typ okna pre extrakciu rysov zvuku je nastavený na obdĺžnikové okno a nie je počítaný koeficient energie. Posun dvoch po sebe nasledujúcich okien je rovný dĺžke okna, čím sú zanedbávané nezrovnalosti na krajoch okien, čo sa v predchádzajúcom experimente ukázalo ako výhodné. V tomto experimente veľkosť okien postupne rastie, až kým nie je posun po sebe nasledujúcich okien približne rovný veľkosti dočasného okna integrácie. Na základe získaných výsledkov (celý súbor získaných dát sa nachádza v prílohe C) bolo možné konštatovať niekoľko faktov:

- V prípade záznamov s pridaným šumom boli získané výsledky zaujímave. Vzdialenosť všetkých nahrávok s hudbou klesala s rastúcou veľkosťou okna. Pre vzdialenosti nahrávok reči vo vzorkovacích frekvenciách 16kHz a 8kHz to platilo tiež. Avšak vzdialenosti týchto nahrávok vo frekvencii 44100Hz so zväčšujúcou sa veľkosťou okna naopak rástli. Toto je ilustrované v tabuľkách 7.12, 7.13 a 7.15. Toto autor opäť pripisuje rozloženiu energie v bielom a ružovom šume.
- s rastúcou veľkosťou okna sa zmenšovala vzdialenosť záznamov so zníženou hlasitosťou,
- s rastúcou veľkosťou okna sa zmenšovala vzdialenosť záznamov zakódovaných do formátu MPEG-1 od originálnych záznamov (tabuľka 7.14),
- s rastúcou veľkosťou okna sa zmenšovala aj vzdialenosť záznamov s rozdielnym zvukom (referenčných a testovacích záznamov, tabuľky 7.16 a 7.17)

Stále však nebol vyriešený problém s malou hodnotou vzdialenosti záznamov hudby so SNR 10dB. Ako vidieť, nie je možné tento problém vyriešiť zmenou veľkosti okna. Skôr naopak, zväčšenie okna vyústilo k zmenšeniu týchto vzdialeností. Túto nevýhodu vyrovná

	zaznam1		zaznam2	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
20	28.49	83.20	734.36	69.10
50	20.04	83.50	830.19	69.00
100	7.81	86.00	1040.27	65.50
200	1.86	82.00	1296.25	62.00

Tabulka 7.12: Vzdialenosti vybraných záznamov s globálnym SNR rovným 30dB (pridaný biely šum, vzorkovacia frekvencia 44100Hz).

	zaznam1		zaznam2	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
20	28.99	84.16	365.96	75.47
50	15.85	82.41	297.33	77.94
100	5.52	75.75	283.37	78.89
200	1.37	75.51	223.23	80.80

Tabulka 7.13: Vzdialenosti vybraných záznamov s globálnym SNR rovným 30dB (pridaný biely šum, vzorkovacia frekvencia 8kHz).

fakt, že zväčšenie veľkosti okna priaznivo vplýva na vzdialenosti záznamov kódovaných do MPEG-1. Avšak nevhodne na vzdialenosti signálov s rôznym zvukom. Preto je nutné vybrať kompromisné riešenie, kde bude zachovaná rovnováha medzi zlepšením výsledkov záznamov vo formáte MPEG-1 a zhoršením výsledkov referenčných a testovacích nahrávok.

V každom prípade zväčšiť okno je vhodné, pretože vzdialenosť záznamov kódovaných do MPEG-1 klesá so zväčšujúcou sa veľkosťou okna rýchlejšie ako klesá vzdialenosť referenčných záznamov. Okno s veľkosťou 200ms ale nie je vhodné, pretože vzdialenosť nahrávky reči so SNR 30dB je väčšia ako vzdialenosť dvoch rockových testovacích záznamov.

Najväčšie prijateľné okno má teda veľkosť 100ms. Táto veľkosť okna bola vybraná ako vhodné nastavenie, keďže takýmto spôsobom bude extrahovaných menej vektorov MFC koeficientov (ako s použitím 50ms okna) a tým sa ušetria výpočtové nároky v samotnom procese diff.

	zaznam1		zaznam2	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
20	650.80	67.20	1177.87	64.69
50	399.49	74.00	921.36	66.91
100	209.74	78.00	693.54	66.83
200	51.74	70.00	557.02	64.64

Tabulka 7.14: Vzdialenosti vybraných záznamov kódovaných do formátu MPEG-1 (vzorkovacia frekvencia 44100Hz).

	zaznam1		zaznam2	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
20	269.28	69.00	1246.09	59.30
50	223.03	67.50	1511.42	59.00
100	140.99	71.00	1877.47	59.50
200	70.87	58.00	2312.48	56.00

Tabulka 7.15: Vzdialenosti vybraných záznamov s globálnym SNR rovným 10dB (vzorkovacia frekvencia 44100Hz).

	zaznam6		zaznam7	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
20	2118.39	58.80	2444.10	55.60
50	2253.13	60.00	2353.15	58.00
100	2141.84	57.00	2045.80	57.00
200	2009.84	58.00	1572.66	58.00

Tabulka 7.16: Vzdialenosti referenčných záznamov (vzorkovacia frekvencia 44100Hz).

	klasika		rock	
dĺžka okna [ms]	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
20	1545.22	55.63	1422.70	57.46
50	1567.46	56.00	1303.77	56.25
100	1522.20	55.67	1186.31	57.00
200	1446.97	56.33	1017.36	55.67

Tabulka 7.17: Vzdialenosti kontrolných záznamov (vzorkovacia frekvencia 44100Hz).

7.2.4 Zmenšenie objemu extrahovaných dát

Po získaní vhodného nastavenia typu okna jeho veľkosti a posunu, je vhodné nejakým spôsobom zredukovať dimenzionalitu extrahovaných dát, pretože počítanie vzdialenosti bodov v dvanásť rozmernom priestore je výpočtovo náročné. Z tohoto dôvodu bol použitý algoritmus SVD k redukcii počtu dimenzií.

Prvý pokus o redukciu bol neúspešný, pretože bola vykonaná okamžite po extrahovaní MFC koeficientov z jedného audio záznamu. Tento prístup pracoval výborne pre testovacie záznamy a ich poškodené verzie s rovnakou dĺžkou. V tomto prípade boli výsledky značne zlepšené. Problém nastal pokiaľ sa porovnávali dvojice iné ako originál a jeho upravená verzia. Dokonca napríklad ak bola porovnávaná dvojica *zaznam3* a *zaznam4* (*zaznam4* bol vytvorený zo záznamu *zaznam3* viď 7.2.1) nebolo správne určené, že stačí pridať úseky ticha, ale že treba zmeniť celý záznam. Vyšlo najavo, že pridanie signálov zvuku k pôvodnému signálu, značne ovplyvňuje algoritmus SVD a pre rovnaké signály sú vypočítané rôzne hodnoty k-aproximácie.

Vyriešenie tohoto problému zabezpečilo až zjednotenie extrahovaných dát zo signálov zvuku oboch porovnávaných záznamov. To znamená, že miesto počítania k-aproximácie z každej matice samostatne, sa vytvorila jedna matica obsahujúca všetky dáta, ktorá bola následne spracovaná algoritmom SVD.

Opäť boli vykonané experimenty, ktoré mali ukázať aký počet koeficientov je dostatočný k aproximácii získaných dát tak, aby neboli znehodnotené vzdialenosti signálov s rôznym zvukom. To znamená, že boli rovnako ako v minulých experimentoch počítané vzdialenosti originálnych nahrávok a ich poškodených verzií a tiež referenčných záznamov. Avšak tento krát pred samotným počítaním vzdialenosti bola vypočítaná k-aproximácia extrahovaných dát. Experiment sledoval akým spôsobom sa menia vzdialenosti nahrávok s rovnakým a rôznym zvukom v závislosti od počtu nový dimenzií vypočítaných algoritmom SVD. Cieľom bolo zistiť aká najmenšia hodnota k k-aproximácie, neznehodnotí dáta získané extrakciou MFC koeficientov.

Úplné výsledky tohoto experimentu sú uvedené v prílohe D. Vo všeobecnosti výsledky experimentov boli očakávané. To znamená, že vzdialenosti, ktoré boli malé, boli aj po použití k-aproximácie dát (namiesto pôvodne extrahovaných dát) malé (napríklad viď tabuľku 7.18, k je nový počet dimenzií dát), pričom boli zachované pomery vzdialeností medzi nahrávkami s rovnakým zvukom a rôznym zvukom. Tento fakt bol zachovaný už pri redukcii na tri dimenzie.

	zaznam1		**zaznam2**	
k	**vzdialenosť**	**počet [%]**	**vzdialenosť**	**počet [%]**
3	0.00	86.00	0.02	80.50
4	0.00	89.00	0.05	82.00
5	0.00	77.00	0.27	88.50
6	0.03	88.00	0.55	81.50

Tabulka 7.18: Vzdialenosti vybraných záznamov s globálnym SNR rovným 30dB (k je počet dimenzií dát, pridaný biely šum, vzorkovacia frekvencia 44100Hz).

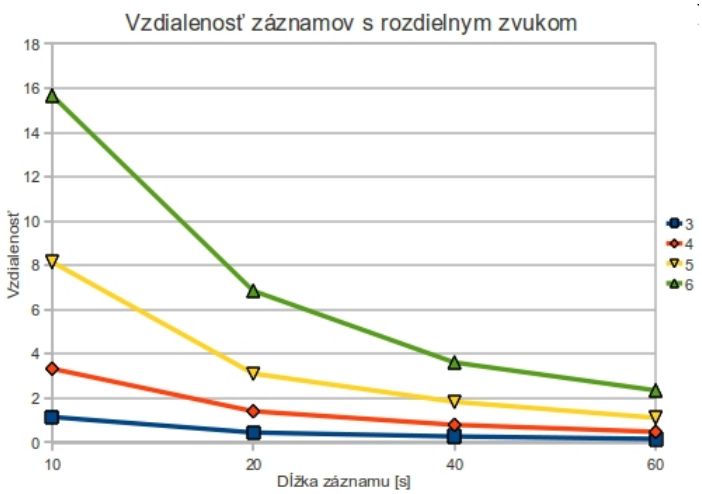
Avšak vo výsledkoch nastal zaujímavý jav, kde sa vzájomná vzdialenosť nahrávok zmenšovala s ich dĺžkou. Toto je badateľné najmä pri vzdialenostiach referenčných a testovacích záznamov, kde je tento jav najviac závažný, keďže je nutné udržať hodnoty ich vzdialeností čo najväčšie. Vzdialenosť týchto záznamov o dĺžke 10s bola podstatne väčšia ako v prípade

nahrávok s dĺžkou 60s. Toto je ilustrované tabuľkou 7.19, kde záznamy zaznam6, zaznam7 majú dĺžku 10s a záznamy zaznam8, zaznam9 majú 60s.

k	zaznam6	zaznam8	zaznam7	zaznam9
3	0.97	0.16	1.23	0.13
4	3.08	0.49	3.17	0.39
5	7.01	1.18	7.27	0.95
6	14.94	2.41	13.85	1.99

Tabulka 7.19: Vzdialenosti vybraných referenčných záznamov (vzorkovacia frekvencia 44100Hz).

Aby bolo tento jav možné sledovať, bol vykonaný experiment, v ktorom sa počítala vzdialenosť dvoch nahrávok s rozdielnym zvukom (jeden obsahoval rock druhý klasickú hudba) s rozdielnymi dĺžkami. Výsledok tohoto experimentu je zobrazený v grafe 7.3. Je vidieť, že klesanie vzdialenosti nie je závislé na počte nových dimenzií a tento pokles vzdialenosti je dosť razantný. Toto je veľmi vážny problém, keďže tento jav znemožňuje určiť pevnú hodnotu hranice zároveň s použitím SVD, ktorá určuje rozdiel medzi tým či sú dva signály rovnaké na základe ich vzájomnej vzdialenosti, pretože sa táto hodnota mení s dĺžkou záznamov.



Obrázek 7.3: Klesanie vzdialenosti záznamov s rozdielnym zvukom v závislosti na ich dĺžke po použití algoritmu SVD (k zodpovedá výslednému počtu dimenzií po redukcii).

V každom prípade je však možné povedať, že až na túto nepríjemnú vlastnosť SVD, je jeho použitie výhodné. So zachovaním pomeru vzdialeností signálov s rovnakým a rôznym zvukom, dokáže znížiť dimenzionalitu extrahovaných dát až štyri krát a tým podstatne zlepšiť výkon aplikácie. S premenlivou vzdialenosťou nahrávok sa bude nutné vysporiadať meniacou sa hodnotou hranice v závislosti od dĺžky porovnávaných záznamov.

Použitím SVD sa opäť nepodarilo vyriešiť problém s malými hodnotami vzdialenosti nahrávok s hudbou obsahujúcimi veľký šum. Toto je spôsobené vlastnosťami MFC koeficientov a tým pádom je nutné vo výslednej aplikácii tento fakt akceptovať.

Kapitola 8

Testy aplikácie

V predchádzajúcich kapitolách bol uvedený návrh a spôsob implementácie aplikácie. Taktiež boli popísané experimenty, ktoré slúžili na určenie vhodných nastavení knižnice pri extrakcii MFC koeficientov a následne boli tieto nastavenia vybrané. Teraz prichádzajú na rad testy výslednej vytvorenej aplikácie a určenie jej možností. Taktiež budú identifikované jej slabé stránky, ktoré by mohli byť vylepšené v ďalších verziách.

8.1 Základné testy

Najprv boli vykonané základné testy na záznamoch rovnakej dĺžky. Jeden záznam bol originál a druhý vznikol určitou zmenou tohoto originálu. Účelom bolo analyzovať schopnosť aplikácie správne určiť, či je celkový signálu zvuku rovnaký alebo nie. V týchto experimentoch budú použité rovnaké vzorky záznamov ako v predchádzajúcej kapitole (viď 7.2.1).

Ako prvý bude porovnávaný záznam hudby s dĺžkou 10s a jeho upravené varianty:

- s pridaným šumom (globálne SNR 30dB)
  `python3.1 soundDiff.py zaznam1 zaznam1SNR30`
  Výstup bol prázdny, čo znamená, že je správny, keďže nie je počuť rozdiel vo zvuku nahrávok. Rovnaký výsledok bol taktiež získaný pre rovnaké záznamy v oboch nižších frekvenciách,

- zakódovaný do MPEG-1
  `python3.1 soundDiff.py zaznam1 zaznam1MPEG`
  Výstup bol opäť prázdny, čo znamená, že je správny keďže nie je počuť rozdiel vo zvuku nahrávok. Rovnaký výsledok sa podarilo získať aj pre tie isté záznamy v oboch nižších frekvenciách,

- so zníženou hlasitosťou
  `python3.1 soundDiff.py zaznam1 zaznam-25dB`
  Výstup bol taktiež opäť prázdny, tým pádom aj správny, pre všetky vzorkovacie frekvencie,

- s pridaným šumom (globálne SNR 10dB):
  `python3.1 soundDiff.py zaznam1 zaznam1SNR10`
  Výstup bol prázdny, čo však teraz znamená, že bol nesprávny, pretože je počuť rozdiel vo zvuku nahrávok, keďže pridaný šum je veľký. Rovnaký výsledok bol taktiež získaný pre rovnaké záznamy v oboch nižších frekvenciách. Problém s ignorovaním veľkého

šumu v nahrávkach s hudbou bol už spomenutý v kapitole 7.2 a je spôsobený vlastnosťami extrahovaných MFC koeficientov. Riešením tohoto problému by bolo nájsť inú reprezentáciu zvuku, ktorá by dokázala lepšie registrovať šum vo zvuku hudby,

- záznam inej hudby s rovnakou dĺžkou:
  `python3.1 soundDiff.py zaznam1 zaznam7 0.00,10.00c0.00,10.00`

  Výstup aplikácie je v tomto prípade správny a podarilo sa ho zopakovať pre všetky ďalšie vzorkovacie frekvencie.

Nasleduje porovnanie záznamu reči s dĺžkou 20s a jeho variantov:

- S pridaným šumom(globálne SNR 30dB)
  ```
  python3.1 soundDiff.py zaznam2 zaznam2SNR30
  0.00,0.40c0.00,0.40
  1.60,2.60c1.60,2.60
  5.10,5.80c5.10,5.80
  8.70,9.40c8.70,9.40
  12.10,13.00c12.10,13.00
  13.50,14.70c13.50,14.70
  15.70,16.80c15.70,16.80
  17.70,20.00c17.70,20.00
  ```

  Zhodnotenie tohoto výsledku je problematické. Po porovnaní výstupu a tvaru vlny zvuku, sa prišlo k záveru, že aplikácia ako rôzne určila časti, kde v originálnom zázname bolo ticho a úseky na začiatku a konci slova. To znamená, že pridaný šum znehodnotil tieto časti obsahujúce menej energie,

- s pridaným šumom(globálne SNR 50dB)
  ```
  python3.1 soundDiff.py zaznam2 zaznam2SNR50
  0.00,0.40c0.00,0.40
  1.80,2.50c1.80,2.50
  12.20,12.70c12.20,12.70
  15.70,16.50c15.70,16.50
  ```

  Je vidieť, že ako odlišné časti bola určená podmnožina z predchádzajúceho testu, keďže bol pridaný menší šum, ktorý znehodnotil len úseky signálu s najmenšou energiou,

- zakódovaný do MPEG-1
  ```
  python3.1 soundDiff.py zaznam2 zaznam2MPEG
  3.80,4.30c3.80,4.30
  7.50,8.10c7.50,8.10
  11.20,11.50c11.20,11.50
  13.50,13.70c13.50,13.70
  13.80,14.50c13.80,14.50
  17.30,18.20c17.30,18.20
  19.00,19.80c19.00,19.80
  19.90,20.00d19.90
  ```

Výstup tohoto testu nie je správny. Zjavne má aplikácia problémy s rečou kódovanou do formátu MPEG-1. Toto je spôsobené nízkou hranicou pri porovnávaní podobnosti dvoch signálov zvuku. Zvýšenie by pomohlo výstup zlepšiť, avšak tým by sa zhoršil výstup porovnania dvoch signálov zvuku hudby, keďže hodnoty extrahovaných MFC koeficientov sú závislé aj na type zvuku (viď kapitolu 7.2.2) a hranice boli po dohode s vedúcim práce nastavené skôr pre hudbu ako pre reč,

- so zníženou hlasitosťou
```
python3.1 soundDiff.py zaznam2 zaznam2-25dB
2.00,2.50c2.00,2.50
12.20,12.70c12.20,12.70
16.00,16.30c16.00,16.30
```

  Aplikácia zle identifikovala približne 1.5 sekundy z celého záznamu, čo nie je zlý výsledok, hoci v prípade porovnania hudby bol ideálny,

- s pridaným šumom (globálne SNR 10dB)
```
python3.1 soundDiff.py zaznam2 zaznam2SNR10
0.00,20.00c0.00,20.00
```

  Výstup je v tomto prípade ideálny. V prípade záznamov v nižšej vzorkovacej frekvencii bol však výsledok značne horší, čo sa pripisuje tomu, že šum má v týchto záznamoch menej energie (viď 7.2.2),

- záznam inej reči s rovnakou dĺžkou
```
python3.1 soundDiff.py zaznam2 zaznam10
0.00,10.00c0.00,10.00
```

  Výstup bol ideálny pre všetky vzorkovacie frekvencie záznamu.

8.2 Ďalšie testy

V tejto časti budú uvedené rozšírené testy výslednej aplikácie, kde bude testovaná správnosť výstupu na zložitejších signáloch zvuku. Bude sledovaná schopnosť aplikácie správne určiť, ktoré časti signálu zvuku boli pridané, vymazané alebo zmenené.

- V prvom teste bol vytvorený audio záznam s rockovou hudbou. Druhý záznam vznikol pridaním približne troch sekúnd klasickej hudby na začiatok pôvodného záznamu a ôsmich sekúnd rockovej hudby jeho koniec.
```
python3.1 soundDiff.py track1 track2
0.00,0.20c0.00,3.30
9.30a12.40,17.10
```

  Výsledok bol správny, až na okraje signálu zvuku, kde dochádzalo k jeho zmene. Toto je spôsobené pomerne veľkým oknom alebo určitou zotrvačnosťou signálu, ktorú prináša porovnanie pomocou DTW, avšak takéto malé odchýlky môžu byť tolerované,

- v druhom teste sa porovnali záznamy v opačnom poradí, čo znamená, že sa hľadal spôsob ako je možné zmeniť druhý záznam na prvý. Takto sa overí, že aplikácie dokáže

identifikovať časti pôvodného záznamu, ktoré boli zmazané. Navyše je takto možné skontrolovať stabilitu výstupu.

```
python3.1 soundDiff.py track2 track1
0.00,3.30c0.00,0.20
12.40,17.10d9.30
```

Výsledok bol správny a je vidieť, že zodpovedá výsledku prvého testu. To jest, ak druhý záznam vznikol pridaním zvuku na začiatok a na koniec prvého záznamu, prvý musí vzniknúť, ak sa tieto pridané časti z druhého záznamu odstránia.

- V ďalšom teste bol vytvorený záznam reči s dĺžkou približne 3 sekundy. Druhý záznam vznikol pridaním piatich sekúnd hudby na začiatok prvej nahrávky a štyroch sekúnd inej reči na koniec tejto nahrávky.

```
python3.1 soundDiff.py track3 track4
0.00,0.40c0.00,6.10
2.90,3.10c8.60,12.80
```

Výsledok bol opäť správny, až na časti kde dochádzalo k zmene signálu. Je si možné všimnúť, že v tomto prípade boli nesprávne identifikované väčšie okrajové časti ako v predchádzajúcom teste.

- V tomto teste bolo opäť prehodené poradie porovnávaných záznamov.

```
python3.1 soundDiff.py track4 track3
0.00,6.10c0.00,0.40
8.60,12.80c2.90,3.10
```

Výsledok bol znovu správny a zodpovedá výsledku získaného v treťom teste.

- V poslednom teste boli nahrávané zvuky z televízie. Mikrofóny boli umiestnené v rôznej vzdialenosti od zdroja. Prvý bol vzdialený asi 0.5 metra. Druhý asi 2m. Bola nahraná minúta zvuku. Následne sa sa kvôli časovej nenáročnosti z tejto minúty vybralo len prvých 10 sekúnd, ktoré sa následne porovnali.

```
python3.1 soundDiff.py track5 track6
0.00,10.00c0.00,10.00
```

Výstup bol v tomto prípade nesprávny. Príčinou môže byť to, že nahrávky sú veľmi nekvalitné z dôvodu použitia nie veľmi kvalitných mikrofónov.

Kapitola 9

Záver

Táto práca sa zaoberala vývojom aplikácie diffu pre audio dokumenty. V úvode boli analyzované informácie o vnímaní zvuku ľudským sluchovým aparátom a uvedené jeho špecifiká a obmedzenia, s ktorými bolo nutné počítať, ak mala byť implementácia cieľovej aplikácie úspešná. V ďalšom kroku sa ako vhodná reprezentácia zvuku vybrali koeficienty cepstra mel-frekvencie. V práci je ďalej podrobne uvedený návrh, implementácia a externé nástroje použité v tejto implementácii.

Následne boli vykonané testy, ktoré mali zabezpečiť identifikáciu vhodných nastavení extrakcie MFC koeficientov zo signálu zvuku. Vybrané nastavenia sa prekvapivo značne líšili od tých, ktoré sú zvyčajne používané v iných oblastiach ako napríklad v rozpoznávaní reči. Po vybraní nastavení sa pristúpilo ku konečnému testovaniu.

Výsledky testov však boli pomerne rozpačité. Problémovým sa ukázali hodnoty MFC koeficientov, ktoré príliš záviseli na type zvuku. Z tohoto dôvodu bolo možné nastaviť aplikáciu tak, aby dávala lepšie výsledky buď pre hudbu alebo reč. Po dohode s vedúcim práce bola aplikácia nastavená tak, aby dávala v prípade porovnávania hudby lepšie výsledky. Bolo ukázané, že sa nástroj v tomto prípade bez problémov vysporiada s kódovaním zvuku, či pridaním rôznych typov šumu. V prípade signálov reči sú výsledky o niečo horšie. Celkovo je však možné hodnotiť implementovanú aplikáciu diffu ako úspešnú.

V prípade pokračovania práce na tomto nástroji, by bolo vhodné vyriešiť problém nízkej rozlišovacej schopnosti šumu vo zvuku hudby a tiež rôznych hodnôt reprezentácie zvuku v závislosti na jeho type. Toto však podľa názoru autora bude možné dosiahnuť len zmenou typu reprezentácie.

Nástroj sa možno stane súčasťou väčšej aplikácie nazývanej MediaDiff, vyvíjanej na Fakulte Informačních technologií Vysokého učení technického v Brne, ktorá bude poskytovať aplikáciu diff pre rôzne typy dokumentov.

Literatura

[1] FFmpeg Documentation [online]. http://www.ffmpeg.org/ffmpeg-doc.html#SEC2, 2010 [cit. 1.1.2011].

[2] Speech Recognition. http://www.learnartificialneuralnetworks.com/speechrecognition.html, 2010 [cit. 6-6-2011].

[3] Hot speech summer in Brno [online]. http://www.signalprocessingsociety.org/technical-committees/list/sl-tc/spl-nl/2010-0 [cit. 22.12.2010].

[4] What is HTK? [online]. http://htk.eng.cam.ac.uk/, [cit. 31.12.2010].

[5] Berry, W.: Computing the SVD. http://web.eecs.utk.edu/ berry/lsi++/node8.html#SECTION00032300000000000000, 1996 [cit. 6-5-2011].

[6] Chandra, K.: *Discriminative Adaptive Training and Bayesian Inference for Speech Recognition*. december 2009.

[7] Cook, P.: *Music, cognition, and computerized sound: an introduction to psychoacoustics*. MIT Press, 2001, iSBN 9780262531900.

[8] Coulter, D.: *Digital Audio Processing*. Focal Press, 2000, iSBN 9780879305666.

[9] Diederich, J.: *Rule extraction from support vector machines*. Springer, 2008, iSBN 9783540753896.

[10] Fastl, H.; Zwicker, E.: *Psychoacoustics: facts and models*. Springer, 2007, iSBN 9783540231592.

[11] Fries, B.; Fries, M.: *Digital audio essentials*. O'Reilly, 2005, iSBN 9780596008567.

[12] Gales, M.; Evermann, G.; Woodland, P.: HTK History [online]. http://htk.eng.cam.ac.uk/docs/history.shtml, 2006 [cit. 22.12.2010].

[13] Garcia, E.: A tutorial on Singular Value Decomposition (SVD) and Latent Semantic Indexing (LSI), its advantages, applications and limitations. http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-1-understand 2007 [cit. 6-5-2011].

[14] Hamawaki, S.; aj.: Feature Analysis and Normalization Approach for Robust Content-Based Music Retrieval to Encoded Audio with Different Bit Rates. In *Advances in multimedia modeling*, Springer Berlin, 2009, s. 298–309.

[15] Harms, D.; McDonnald, K.: *Začínáme programovat v jazyce Python.* Computer Press, 2006, iSBN 80-7226-799-X.

[16] Hartmann, M.: *Signals, sound, and sensation. American Institute of Physics.* American Institute of Physics, 1997, iSBN ISBN 9781563962837.

[17] Heckel, P.: A technique for Isolating Differences Between Files. *Communications of the ACM*, apríl 1978.

[18] Hoffman, M.: An Introduction to Fourier Theory. `http://gershwin.ens.fr/vdaniel/Doc-Locale/Cours-Mirrored/Maths-Stuff/fourier/`, 1997 [cit. 22-12-2010].

[19] Howard, D.; Angus, J.: *Acoustics and psychoacoustics.* Focal Press, 2009, iSBN 9780240521756.

[20] Hunt, J.; McIlroy, M.: An Algorithm for Differential File Comparison. Technická zpráva, Bell Laboratories, 1976.

[21] Ikeda, A.; Inoué, Y.: *Event-related potentials in patients with epilepsy : from current state to future prospects.* John Libbey Eurotext, 2008, iSBN 9782742006847.

[22] Lagrange, M.; Raspaud, M.; Badeau, R.; aj.: Explicit modeling of temporal dynamics within musical signals for acoustical unit similarity. *Pattern Recognition Letters*, 2010: s. 1498–1506.

[23] Lyu, Y.; aj.: Coherent Bag-of Audio Words Model For Efficient Large-Scale Video Copy Detection. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, 2010.

[24] Mallat, S.: *A wavelet tour of signal processing: the Sparse way.* Academic Press, 2009, iSBN 9780123743701.

[25] Miller, W.; Myers, W.: A File Comparison Program. *Software. Practice and Experience*, november 1985.

[26] MobileReference: *Signal Processing Quick Study Guide for Smartphones and Mobile Devices.* MobileReference, 2007, iSBN 9781605010670.

[27] Myers, W.: An O(ND) Difference Algorithm and Its Variations. Technická zpráva, Department of Computer Science, University of Arizona, 1986.

[28] Norton, R.: Diff Algorithm [online]. `http://c2.com/cgi/wiki?DiffAlgorithm`, 2010 [cit. 22.12.2010].

[29] Oualline, S.: *Practical C++ programming.* O'Reilly, 2003, iSBN 9780596004194.

[30] Paliwal, K.: On the use of filter-bank energies as features for robust speech recognition. In *Signal Processing and Its Applications, 1999. ISSPA '99. Proceedings of the Fifth International Symposium on*, 1999, s. 641 –644 vol.2.

[31] Park, T.: *Introduction to digital signal processing: computer musically speaking.* World Scientific, 2010, iSBN 9789812790279.

[32] Prasad, B.; Prasanna, S.: *Speech, Audio, Image and Biomedical Signal Processing Using Neural Networks*. Springer, 2008, iSBN 9783540753971.

[33] P.Sinha: *Speech Processing in Embedded Systems*. Springer, 2010, iSBN 9780387755809.

[34] Salvatore, K.: *Maximum PC*, September 2001: str. 53.

[35] Taymans, W.; aj.: GStreamer Application Development Manual [online]. `http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/index.html`, [cit. 1.1.2011].

[36] Tichy, F.: The String-to-String Correction Problem with Block Moves. *ACM Transactions on Computer Systems*, 1984.

[37] Tsihrintzis, G.; Jain, L.: *Multimedia Services in Intelligent Environments: Advanced Tools and Methodologies Studies in Computational Intelligence*. Springer, 2008, iSBN 9783540784913.

[38] Tzanetakis, G.; Cook, P.: Musical Genre Classification of Audio Signals. In *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*, júl 2002.

[39] Vaseghi, S.: *Advanced digital signal processing and noise reduction*. John Wiley, 2000, iSBN 9780471626923.

[40] Watkinson, J.: *An introduction to digital audio*. Focal Press, 2002, iSBN 9780240516431.

[41] Zhang, T.; Kuo, J.: *Content-based audio classification and retrieval for audiovisual data parsing*. Springer, 2001, iSBN 9780792372875.

[42] Zölzer, U.: *Digital audio signal processing*. Wiley, 2008, iSBN 9780470997857.

[43] Özer, H.; aj.: Perceptual Audio Hashing Functions. *EURASIP Journal on Applied Signal Processing*, 2005.

Příloha A

Návod na použite a obsah CD

V tejto časti je uvedený obsah CD a návod na inštaláciu a spustenie vytvorenej aplikácie.

Na CD sa v zložke `sources` nachádzajú zdrojové kódy aplikácie. V zložke `data` sa nachádzajú dáta z testovaní vo forme html tabuliek. Zložka `Ukazky` obsahuje skript `test.sh`, ktorý ukáže funkčnosť aplikácie na audio súboroch priložených v zložke `testData`. Potrebné informácie k spusteniu skriptu sú uvedené v `README`. Inštalácia programu je jednoduchá. Stačí ako administrátor spustiť inštalačný skript. Vopred však treba mať nainštalované knižnice `CLAPACK` a `LIBLAPACK`. Inštalačný skript sa tiež pokúsi o inštaláciu potrebných nástrojov gstreameru.

Spustenie aplikácie sa udeje spustením modulu `soundDiff.py` v interprete pythonu verzie 3. Ukážka:
`python3.1 soundDiff.py zaznam1 zaznam2 [parametre]`
Parametre:
`-t číslo` nastavuje hodnotu hranice pri druhom porovnávaní na hodnotu `číslo`.
`--treshold číslo` nastavuje hodnotu hranice pri druhom porovnávaní na hodnotu `číslo`.
`-s` zabráni použitiu SVD pri porovnávaní.
`--disable-svd` zabráni použitiu SVD pri porovnávaní.

Príloha B

Dáta experimentov s typom okna

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	28.494	83.200	28.498	83.200	734.361	69.100	737.503	69.400
hammingovo	60.009	81.982	60.013	81.982	2275.778	57.029	2278.913	57.029
hannovo	63.919	82.282	63.922	82.282	7963.745	37.669	7966.880	37.719
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	19.887	85.400	19.891	85.400	86.132	71.925	157.822	73.250
hammingovo	42.781	84.181	42.786	84.181	131.805	70.059	203.469	71.434
hannovo	45.757	84.314	45.762	84.314	139.060	70.034	210.723	71.259
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	251.792	76.178	311.112	68.594				
hammingovo	639.709	78.503	698.981	72.512				
hannovo	1847.717	80.216	1906.989	80.216				

Tabulka B.1: Vzdialenosti záznamov s prianým bielym šumom od originálov. Globálne SNR rovné 30dB. Vzorkovacia frekvencia 44100Hz.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	650.801	67.200	652.021	67.400	1177.878	64.694	1181.198	64.694
hammingovo	995.102	64.765	996.293	64.965	1876.424	64.293	1879.708	64.293
hannovo	1035.551	65.265	1036.742	65.265	2054.686	64.995	2057.970	65.095
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	588.473	67.790	589.260	67.790	441.958	65.415	442.963	65.415
hammingovo	902.907	66.183	903.677	66.149	678.183	62.938	679.162	62.963
hannovo	943.688	66.199	944.458	66.233	709.408	62.625	710.387	62.650
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	583.820	68.292	585.645	68.391				
hammingovo	916.530	67.122	918.360	67.172				
hannovo	980.728	68.411	982.559	68.480				

Tabulka B.2: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 44100Hz.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	0.027	92.800	33.163	92.600	150.548	81.100	183.485	81.100
hammingovo	0.081	90.991	33.218	90.991	420.707	72.686	453.653	72.686
hannovo	0.096	91.091	33.232	91.091	2646.196	41.321	2679.141	41.371
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	0.016	93.500	33.153	93.400	0.012	93.625	24.873	24.975
hammingovo	0.061	93.666	33.197	93.649	0.045	94.224	24.909	24.966
hannovo	0.073	93.766	33.210	93.816	0.055	94.512	24.919	24.966
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	33.037	95.663	59.251	95.505				
hammingovo	84.112	92.069	110.332	91.900				
hannovo	521.286	80.236	547.506	80.236				

Tabulka B.3: Vzdialenosti záznamov so zníženou hlasitosťou o 25dB od originálov. Vzorkovacia frekvencia 44100Hz.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	269.280	69.000	269.489	69.000	1246.094	59.300	1259.216	58.200
hammingovo	469.050	67.167	469.259	67.167	3644.137	59.430	3657.230	59.430
hannovo	486.376	67.067	486.585	67.067	9975.323	41.521	9988.416	41.571
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	215.674	70.767	215.871	70.800	237.659	65.825	353.748	60.925
hammingovo	364.884	70.728	365.083	70.728	380.040	67.508	496.091	56.932
hannovo	379.887	70.712	380.087	70.712	396.053	67.583	512.105	56.795
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	502.116	76.198	600.979	62.713				
hammingovo	1137.828	76.087	1236.619	76.255				
hannovo	2443.368	81.295	2542.160	81.374				

Tabulka B.4: Vzdialenosti záznamov s prianým ružovým šumom od originálov. Globálne SNR rovné 10dB. Vzorkovacia frekvencia 44100Hz.

	zaznam6				zaznam7			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	2118.397	58.800	2140.801	58.800	2444.106	55.600	2445.918	55.600
hammingovo	3371.741	59.960	3394.156	59.760	3236.033	58.358	3237.870	58.358
hannovo	3536.222	58.959	3558.636	59.059	3288.104	58.058	3289.942	58.058
	zaznam8				zaznam9			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	2346.595	57.167	2369.600	57.167	2361.909	57.033	2363.563	57.000
hammingovo	3583.962	57.193	3606.975	57.243	3035.604	58.443	3037.269	58.426
hannovo	3744.992	57.176	3768.005	57.210	3087.793	58.510	3089.458	58.560
	zaznam10							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	3975.666	60.561	4005.453	60.160				
hammingovo	6482.372	58.258	6512.149	58.008				
hannovo	7759.547	55.455	7789.324	55.405				

Tabulka B.5: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 44100Hz.

	klasika				rock			
	bez energie		s energiu		bez energie		s energiu	
Typ okna	vzdialenosť	počet	vzdialenosť	počet	vzdialenosť	počet	vzdialenosť	počet
obdĺžnikové	1545.225	55.633	1547.858	55.633	1422.706	57.467	1423.367	57.400
hammingovo	2180.175	55.626	2182.800	55.526	1761.268	57.510	1761.928	57.510
hannovo	2279.732	55.809	2282.358	55.843	1795.698	57.560	1796.358	57.526

Tabulka B.6: Vzdialenosti kontrolných záznamov.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	28.998	84.168	29.002	84.168	365.966	75.475	369.130	75.475
hammingovo	72.660	80.060	72.664	80.060	671.176	66.817	674.332	66.917
hannovo	78.924	80.261	78.928	80.261	807.441	63.664	810.597	63.714
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	20.023	85.962	20.028	85.962	92.175	71.968	155.471	73.143
hammingovo	50.837	83.978	50.842	83.978	156.405	70.043	219.722	70.943
hannovo	55.674	84.045	55.679	84.045	168.763	70.018	232.079	70.780
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	163.723	71.816	216.036	71.361				
hammingovo	296.020	68.499	348.375	66.102				
hannovo	337.408	69.489	389.764	65.855				

Tabulka B.7: Vzdialenosti záznamov s prianým bielym šumom od originálov. Globálne SNR rovné 30dB. Vzorkovacia frekvencia 16kHz.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	1412.737	59.073	1416.884	59.073	2130.568	57.573	2141.963	57.472
hammingovo	2103.606	59.879	2107.485	59.778	3471.494	56.849	3482.869	56.799
hannovo	2162.936	59.577	2166.815	59.577	3721.545	56.648	3732.920	56.397
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	1297.669	60.827	1299.926	60.727	974.993	58.684	977.636	58.659
hammingovo	1887.643	61.238	1889.844	61.238	1417.195	58.053	1419.792	57.990
hannovo	1944.895	61.238	1947.096	61.138	1462.016	57.878	1464.612	57.840
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	1203.727	60.729	1208.838	60.590				
hammingovo	1828.423	61.419	1833.563	61.389				
hannovo	1914.034	61.835	1919.173	61.775				

Tabulka B.8: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 16kHz.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	298.000	68.337	298.257	68.337	783.740	59.960	798.383	59.359
hammingovo	558.087	65.731	558.350	65.631	1646.429	58.559	1661.038	58.208
hannovo	582.507	65.932	582.770	65.932	1871.923	59.710	1886.532	59.560
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	229.573	69.990	229.826	70.023	239.863	64.341	344.792	62.316
hammingovo	421.408	69.090	421.666	69.090	417.034	65.004	521.983	58.327
hannovo	442.438	69.040	442.697	69.040	440.080	65.079	545.028	58.352
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	380.195	69.954	469.783	58.289				
hammingovo	718.606	70.370	808.269	62.389				
hannovo	781.743	71.262	871.407	64.458				

Tabulka B.9: Vzdialenosti záznamov s prianým ružovým šumom od originálov. Globálne SNR rovné 10dB. Vzorkovacia frekvencia 16kHz.

	zaznam6				zaznam7			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	2364.800	57.315	2387.032	57.715	2439.703	56.313	2441.542	56.513
hammingovo	3741.983	58.717	3764.219	59.018	3365.744	57.515	3367.611	57.515
hannovo	3885.410	58.717	3907.645	58.918	3420.490	57.715	3422.357	57.715
	zaznam8				zaznam9			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	2605.821	55.452	2628.657	55.418	2364.845	57.119	2366.549	57.119
hammingovo	3881.111	56.719	3903.953	56.736	3130.832	57.886	3132.547	57.869
hannovo	4015.722	56.669	4038.564	56.686	3185.822	57.803	3187.538	57.903
	zaznam10							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	3587.019	56.957	3616.899	56.156				
hammingovo	6118.190	56.256	6148.064	55.956				
hannovo	6517.163	56.507	6547.037	56.406				

Tabulka B.10: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 16kHz.

	klasika				rock			
	bez energie		s energiu		bez energie		s energiu	
Typ okna	vzdialenosť	počet	vzdialenosť	počet	vzdialenosť	počet	vzdialenosť	počet
obdĺžnikové	2013.934	55.485	2016.471	55.352	1703.691	56.852	1704.362	56.819
hammingovo	2923.906	56.619	2926.434	56.619	2181.686	57.603	2182.357	57.603
hannovo	3016.245	56.719	3018.774	56.719	2224.932	57.536	2225.602	57.519

Tabulka B.11: Vzdialenosti kontrolných záznamov.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	29.511	81.563	29.516	81.563	384.454	74.274	387.664	74.374
hammingovo	80.792	79.259	80.797	79.259	689.823	69.369	693.020	69.469
hannovo	89.896	79.359	89.900	79.359	795.966	67.718	799.163	67.718
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	19.967	86.729	19.972	86.729	116.549	72.593	174.438	73.343
hammingovo	52.764	84.478	52.769	84.478	204.616	71.380	262.520	71.818
hannovo	58.454	84.545	58.460	84.545	220.355	71.205	278.259	71.555
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	192.794	71.994	240.668	71.163				
hammingovo	345.285	69.390	393.189	68.122				
hannovo	383.188	69.479	431.092	67.835				

Tabulka B.12: Vzdialenosti záznamov s prianým bielym šumom od originálov. Globálne SNR rovné 30dB. Vzorkovacia frekvencia 8kHz.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	1980.254	58.918	1985.989	59.118	2784.487	57.905	2803.184	57.704
hammingovo	3039.023	59.218	3044.593	59.319	4758.290	57.301	4776.897	57.251
hannovo	3110.949	58.918	3116.519	58.918	4979.011	58.107	4997.618	58.056
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	1935.643	57.773	1938.535	57.740	1454.241	54.233	1457.836	54.259
hammingovo	2832.388	58.442	2835.259	58.459	2128.600	54.203	2132.225	54.128
hannovo	2898.372	58.241	2901.243	58.175	2180.315	54.065	2183.940	54.002
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	1738.873	55.800	1747.117	55.701				
hammingovo	2707.089	58.031	2715.422	58.011				
hannovo	2794.609	58.269	2802.941	58.309				

Tabulka B.13: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 8kHz.

	zaznam1				zaznam2			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	311.876	69.339	312.187	69.539	746.101	59.359	760.566	58.859
hammingovo	616.764	67.535	617.086	67.635	1531.186	57.708	1545.591	57.558
hannovo	648.227	67.836	648.549	67.836	1684.043	58.058	1698.448	57.958
	zaznam3				zaznam4			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	238.437	71.957	238.706	71.991	275.097	64.116	372.809	63.691
hammingovo	449.498	71.407	449.770	71.407	491.319	64.266	589.053	60.653
hannovo	471.929	71.691	472.201	71.707	517.822	64.204	615.556	60.528
	zaznam5							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	431.903	68.766	516.909	59.972				
hammingovo	817.114	68.984	902.183	62.181				
hannovo	871.414	69.469	956.483	63.052				

Tabulka B.14: Vzdialenosti záznamov s prianým ružovým šumom od originálov. Globálne SNR rovné 10dB. Vzorkovacia frekvencia 8kHz.

	zaznam6				zaznam7			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	2879.869	55.311	2901.962	55.511	2751.356	55.511	2753.201	55.711
hammingovo	4292.090	56.814	4314.190	56.914	3901.834	56.814	3903.712	56.613
hannovo	4337.992	57.214	4360.093	57.415	3951.744	56.513	3953.622	56.513
	zaznam8				zaznam9			
	bez energie		s energiou		bez energie		s energiou	
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]	vzdialenosť	počet [%]
obdĺžnikové	3211.954	55.285	3234.609	55.252	2601.556	56.886	2603.304	56.852
hammingovo	4574.850	56.135	4597.512	56.085	3600.069	58.903	3601.827	58.903
hannovo	4612.301	56.102	4634.962	56.102	3654.090	58.686	3655.849	58.670
	zaznam10							
	bez energie		s energiou					
Typ okna	vzdialenosť	počet [%]	vzdialenosť	počet [%]				
obdĺžnikové	3298.759	55.656	3330.256	55.656				
hammingovo	5825.334	58.358	5856.807	58.659				
hannovo	6070.986	58.408	6102.459	58.458				

Tabulka B.15: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 8kHz.

	klasika				rock			
	bez energie		s energiu		bez energie		s energiu	
Typ okna	vzdialenosť	počet	vzdialenosť	počet	vzdialenosť	počet	vzdialenosť	počet
obdĺžnikové	2970.161	56.352	2972.635	56.385	2230.749	56.952	2231.446	56.952
hammingovo	4384.069	58.269	4386.535	58.286	2994.613	58.103	2995.309	58.103
hannovo	4410.519	57.786	4412.985	57.836	3051.632	57.986	3052.328	58.003

Tabulka B.16: Vzdialenosti kontrolných záznamov.

Príloha C

Dáta experimentov s dĺžkou okna

	zaznam1		zaznam2		zaznam3		zaznam4	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	28.494	83.200	734.361	69.100	19.887	85.400	86.132	71.925
50	20.049	83.500	830.190	69.000	14.134	85.917	58.084	71.500
100	7.818	86.000	1040.279	65.500	5.769	83.667	44.747	73.500
200	1.866	82.000	1296.253	62.000	1.317	79.667	37.799	75.000
	zaznam5							
dĺžka okna[ms]	vzdialenosť	počet[%]						
20	251.792	76.178						
50	267.082	85.198						
100	294.919	85.050						
200	365.278	82.178						

Tabulka C.1: Vzdialenosti záznamov s prianým bielym šumom od originálov. Globálne SNR rovné 30dB. Vzorkovacia frekvencia 44100Hz.

	zaznam1		zaznam2		zaznam3		zaznam4	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	650.801	67.200	1177.878	64.694	588.473	67.790	441.958	65.415
50	399.497	74.000	921.362	66.917	325.863	72.871	246.519	71.026
100	209.745	78.000	693.546	66.834	168.345	73.790	131.683	74.593
200	51.743	70.000	557.027	64.646	53.964	72.910	50.641	78.697
	zaznam5							
dĺžka okna[ms]	vzdialenosť	počet[%]						
20	583.820	68.292						
50	384.930	74.083						
100	256.580	75.917						
200	157.973	81.548						

Tabulka C.2: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 44100Hz.

	zaznam1		zaznam2		zaznam3		zaznam4	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	269.280	69.000	1246.094	59.300	215.674	70.767	237.659	65.825
50	223.030	67.500	1511.427	59.000	177.994	72.250	188.519	70.938
100	140.994	71.000	1877.477	59.500	117.616	68.667	133.863	64.375
200	70.877	58.000	2312.489	56.000	61.429	64.333	85.175	54.000
	zaznam5							
dĺžka okna[ms]	vzdialenosť	počet[%]						
20	502.116	76.198						
50	547.642	76.386						
100	583.562	79.208						
200	676.951	80.000						

Tabulka C.3: Vzdialenosti záznamov s prianým ružovým šumom od originálov. Globálne SNR rovné 10dB. Vzorkovacia frekvencia 44100Hz.

	zaznam6		zaznam7		zaznam8		zaznam9	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	2118.397	58.800	2444.106	55.600	2346.595	57.167	2361.909	57.033
50	2253.137	60.000	2353.159	58.000	2468.825	57.917	2278.075	59.250
100	2141.849	57.000	2045.803	57.000	2444.059	55.000	2024.743	58.667
200	2009.849	58.000	1572.662	58.000	2363.256	54.667	1651.406	58.333
	zaznam10							
dĺžka okna[ms]	vzdialenosť	počet[%]						
20	3975.666	60.561						
50	4520.071	60.150						
100	4669.626	57.286						
200	4291.508	58.586						

Tabulka C.4: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 44100Hz.

	klasika		rock	
dĺžka okna [ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	1545.225	55.633	1422.706	57.467
50	1567.461	56.000	1303.777	56.250
100	1522.204	55.667	1186.312	57.000
200	1446.976	56.333	1017.360	55.667

Tabulka C.5: Vzdialenosti kontrolných záznamov. Vzorkovacia frekvencia 44100.

	zaznam1		zaznam2		zaznam3		zaznam4	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	28.998	84.168	365.966	75.475	20.023	85.962	92.175	71.968
50	15.854	82.412	297.335	77.945	13.576	86.322	53.157	71.169
100	5.524	75.758	283.378	78.894	5.468	83.306	36.266	73.467
200	1.374	75.510	223.230	80.808	1.159	76.254	28.255	74.937

	zaznam5	
dĺžka okna[ms]	vzdialenosť	počet[%]
20	163.723	71.816
50	123.015	71.917
100	96.447	72.448
200	82.640	72.421

Tabulka C.6: Vzdialenosti záznamov s prianým bielym šumom od originálov. Globálne SNR rovné 30dB. Vzorkovacia frekvencia 16kHz.

	zaznam1		zaznam2		zaznam3		zaznam4	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	1412.737	59.073	2130.568	57.573	1297.669	60.827	974.993	58.684
50	1165.846	60.606	2081.370	57.035	1076.861	63.272	810.204	62.078
100	756.142	61.616	1634.997	59.296	662.847	65.776	500.089	62.578
200	209.411	57.143	863.595	66.667	253.472	61.538	197.080	64.160

	zaznam5	
dĺžka okna[ms]	vzdialenosť	počet[%]
20	1203.727	60.729
50	1069.756	63.677
100	735.227	67.691
200	381.256	72.421

Tabulka C.7: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 16kHz.

	zaznam1		zaznam2		zaznam3		zaznam4	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	298.000	68.337	783.740	59.960	229.573	69.990	239.863	64.341
50	236.799	65.327	760.545	62.155	182.314	69.892	174.465	65.041
100	144.836	67.677	744.319	61.307	118.887	68.114	119.938	60.200
200	68.661	57.143	621.237	61.616	58.181	63.211	74.418	57.895
	zaznam5							
dĺžka okna[ms]	vzdialenosť	počet[%]						
20	380.195	69.954						
50	341.699	74.195						
100	298.808	76.809						
200	260.549	81.349						

Tabulka C.8: Vzdialenosti záznamov s prianým ružovým šumom od originálov. Globálne SNR rovné 10dB. Vzorkovacia frekvencia 16kHz.

	zaznam6		zaznam7		zaznam8		zaznam9	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	2364.800	57.315	2439.703	56.313	2605.821	55.452	2364.845	57.119
50	2511.609	56.281	2339.255	56.281	2726.162	56.547	2274.118	58.048
100	2368.567	56.566	2041.284	55.556	2674.325	55.593	2035.230	58.431
200	2174.735	53.061	1570.159	55.102	2530.270	54.515	1651.933	59.197
	zaznam10							
dĺžka okna[ms]	vzdialenosť	počet[%]						
20	3587.019	56.957						
50	3990.272	57.895						
100	4236.950	56.784						
200	3992.781	57.576						

Tabulka C.9: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 16kHz.

	klasika		rock	
dĺžka okna [ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	2013.934	55.485	1703.691	56.852
50	2076.163	55.880	1572.778	58.215
100	2023.171	55.092	1442.125	56.594
200	1926.606	55.518	1262.034	56.856

Tabulka C.10: Vzdialenosti kontrolných záznamov. Vzorkovacia frekvencia 16000.

	zaznam1		zaznam2		zaznam3		zaznam4	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	1980.254	58.918	2784.487	57.905	1935.643	57.773	1454.241	54.233
50	1842.284	58.794	2858.245	56.927	1794.218	58.110	1354.456	56.516
100	1466.998	58.586	2678.965	56.566	1456.443	58.863	1104.063	56.391
200	747.941	61.224	2030.688	57.576	788.524	60.870	607.672	59.649

	zaznam5	
dĺžka okna[ms]	vzdialenosť	počet[%]
20	1738.873	55.800
50	1686.819	57.957
100	1456.726	60.020
200	922.492	64.484

Tabulka C.11: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 8kHz.

	zaznam6		zaznam7		zaznam8		zaznam9	
dĺžka okna[ms]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
20	2879.869	55.311	2751.356	55.511	3211.954	55.285	2601.556	56.886
50	2998.435	54.271	2689.053	57.789	3382.328	55.630	2533.927	57.631
100	2895.816	55.556	2426.766	53.535	3314.598	53.422	2321.934	56.761
200	2574.363	53.061	1977.869	57.143	3081.371	52.843	1927.953	58.528

	zaznam10	
dĺžka okna[ms]	vzdialenosť	počet[%]
20	3298.759	55.656
50	3643.957	56.140
100	3945.132	54.774
200	3800.269	57.576

Tabulka C.12: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 8kHz.

Příloha D

Dáta experimentov s k-aproximáciou

	zaznam1		zaznam2		zaznam3		zaznam4	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.000	86.000	0.017	80.500	0.000	86.333	0.000	82.125
4	0.004	89.000	0.048	82.000	0.000	88.500	0.001	85.750
5	0.001	77.000	0.274	88.500	0.001	86.667	0.003	84.250
6	0.025	88.000	0.554	81.500	0.002	87.833	0.018	82.500

	zaznam5	
k	vzdialenosť	počet[%]
3	0.003	84.951
4	0.004	82.970
5	0.058	84.257
6	0.127	84.257

Tabulka D.1: Vzdialenosti záznamov s prianým bielym šumom od originálov. Globálne SNR rovné 30dB. Vzorkovacia frekvencia 44100Hz. K určuje k-aproximáciu.

	zaznam1		zaznam2		zaznam3		zaznam4	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.037	78.000	0.036	74.372	0.012	81.636	0.004	79.600
4	0.186	82.000	0.106	73.869	0.035	79.800	0.013	80.726
5	0.427	72.000	0.456	71.357	0.113	80.634	0.040	79.725
6	0.896	70.000	1.052	71.859	0.225	78.965	0.107	81.352

	zaznam5	
k	vzdialenosť	počet[%]
3	0.005	82.755
4	0.015	81.368
5	0.050	80.377
6	0.112	81.467

Tabulka D.2: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 44100Hz. K určuje k-aproximáciu.

	zaznam6		zaznam7		zaznam8		zaznam9	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.974	69.000	1.227	67.000	0.161	68.167	0.129	70.000
4	3.083	68.000	3.172	71.000	0.490	69.833	0.389	70.167
5	7.007	72.000	7.265	70.000	1.180	72.167	0.945	69.667
6	14.943	74.000	13.847	73.000	2.412	70.000	1.992	69.167

	zaznam10	
k	vzdialenosť	počet[%]
3	0.400	67.839
4	1.142	69.849
5	2.892	69.347
6	6.441	65.829

Tabulka D.3: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 44100Hz. K určuje k-aproximáciu.

	klasika		rock	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.132	66.333	0.146	66.833
4	0.439	68.500	0.497	69.833
5	1.008	67.000	1.138	68.833
6	2.094	67.333	2.302	69.667

Tabulka D.4: Vzdialenosti kontrolných záznamov.Vzorkovacia frekvencia 44100.K určuje k-aproximáciu.

	zaznam1		zaznam2		zaznam3		zaznam4	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.000	73.737	0.011	87.437	0.000	89.649	0.000	81.852
4	0.001	82.828	0.067	87.437	0.000	88.481	0.001	85.857
5	0.003	83.838	0.104	86.935	0.000	83.639	0.004	84.606
6	0.016	84.849	0.190	85.427	0.001	84.641	0.012	83.229
	zaznam5							
k	vzdialenosť	počet[%]						
3	0.001	87.116						
4	0.006	88.702						
5	0.009	84.044						
6	0.022	85.927						

Tabulka D.5: Vzdialenosti záznamov s prianým bielym šumom od originálov. Globálne SNR rovné 30dB. Vzorkovacia frekvencia 16kHz. K určuje k-aproximáciu.

	zaznam1		zaznam2		zaznam3		zaznam4	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.252	68.687	0.147	73.367	0.036	73.790	0.012	74.969
4	0.843	71.717	0.446	72.864	0.117	70.284	0.064	73.717
5	2.119	75.758	1.230	73.367	0.307	70.284	0.145	73.968
6	4.450	75.758	2.545	73.869	0.714	69.449	0.327	75.720
	zaznam5							
k	vzdialenosť	počet[%]						
3	0.016	79.782						
4	0.069	77.106						
5	0.143	75.421						
6	0.354	76.611						

Tabulka D.6: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 16kHz. K určuje k-aproximáciu.

	zaznam6		zaznam7		zaznam8		zaznam9	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.986	72.727	0.774	67.677	0.163	68.447	0.182	69.115
4	2.541	68.687	2.521	68.687	0.449	66.945	0.528	68.447
5	6.138	66.667	6.002	67.677	1.135	65.609	1.248	70.284
6	12.680	67.677	13.193	69.697	2.369	67.446	2.402	71.452
	zaznam10							
k	vzdialenosť	počet[%]						
3	0.409	71.859						
4	1.249	67.337						
5	2.988	69.347						
6	6.429	68.844						

Tabulka D.7: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 16kHz. K určuje k-aproximáciu.

	klasika		rock	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.130	66.277	0.152	70.117
4	0.428	68.447	0.515	66.778
5	1.009	66.778	1.181	67.780
6	2.103	64.274	2.375	66.778

Tabulka D.8: Vzdialenosti kontrolných záznamov.Vzorkovacia frekvencia 16000.K určuje k-aproximáciu.

	zaznam1		zaznam2		zaznam3		zaznam4	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.539	69.697	0.276	66.162	0.053	69.064	0.035	74.060
4	1.765	73.737	0.791	64.647	0.252	67.391	0.110	74.311
5	4.410	72.727	1.973	66.162	0.789	68.227	0.263	74.561
6	8.841	75.758	3.946	66.162	1.544	67.726	0.630	72.556
	zaznam5							
k	vzdialenosť	počet[%]						
3	0.037	71.329						
4	0.141	71.528						
5	0.342	73.214						
6	0.709	72.718						

Tabulka D.9: Vzdialenosti záznamov zakódovaných do MPEG-1 od originálov. Vzorkovacia frekvencia 8kHz. K určuje k-aproximáciu.

	zaznam1		zaznam2		zaznam3		zaznam4	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.032	76.768	0.078	65.327	0.005	76.461	0.006	73.968
4	0.168	78.788	0.262	71.859	0.022	76.962	0.014	72.591
5	0.422	77.778	0.742	70.854	0.081	77.796	0.029	73.842
6	0.878	71.717	1.475	74.874	0.195	79.800	0.067	73.091

	zaznam5	
k	vzdialenosť	počet[%]
3	0.005	81.566
4	0.019	77.304
5	0.047	77.007
6	0.093	79.980

Tabulka D.10: Vzdialenosti záznamov s prianým ružovým šumom od originálov. Globálne SNR rovné 10dB. Vzorkovacia frekvencia 8kHz. K určuje k-aproximáciu.

	zaznam6		zaznam7		zaznam8		zaznam9	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	1.140	63.636	1.007	63.636	0.151	66.277	0.158	67.446
4	3.252	69.697	3.232	61.616	0.488	66.778	0.477	66.444
5	7.314	63.636	7.675	65.657	1.037	66.611	1.182	67.613
6	15.502	66.667	14.193	62.626	2.283	66.778	2.564	67.112

	zaznam10	
k	vzdialenosť	počet[%]
3	0.421	66.332
4	1.321	67.337
5	3.366	66.332
6	6.653	66.332

Tabulka D.11: Vzdialenosti referenčných záznamov. Vzorkovacia frekvencia 8kHz. K určuje k-aproximáciu.

	klasika		rock	
k	vzdialenosť	počet[%]	vzdialenosť	počet[%]
3	0.124	70.618	0.152	70.618
4	0.379	69.616	0.453	69.783
5	0.963	67.446	1.050	71.619
6	2.055	70.618	2.073	67.780

Tabulka D.12: Vzdialenosti kontrolných záznamov.Vzorkovacia frekvencia 8000.K určuje k-aproximáciu.